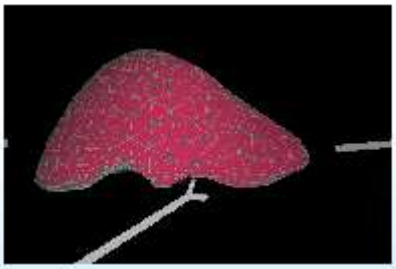
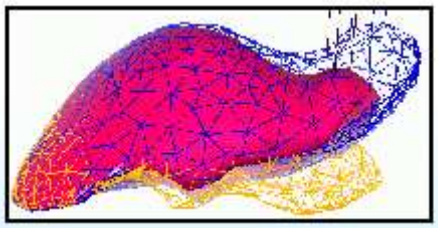
Optimisation de modèles physiques de tissus mous pour la simulation chirurgicale

Cédric CHECOURY

Rapport de stage (septembre 2002) pour :
Le DEA SIC Images - Vision
et
La troisième année d'EURECOM





Stage dirigé par Nicholas Ayache et Hervé Delingette

Linear Elastic Models Optimization for Soft Tissues in the Real-Time Surgery Simulation.

Abstract: In this presentation, we focus on a physically modeling realistic behavior of soft tissues, within the framework of real-time surgery simulation. We describe two main models of the simulator, which are based on linear elasticity theory and finite elements modeling.

The first model (the "pre-computed" model) pre-computes the deformations and forces applied on a finite element model, allowing the deformation of large structures in real-time. The problem with this numerical approach is that it does not allow any topology changes of the mesh. So that cutting the mesh during the surgery simulation becomes impossible.

The second physical model (the "mass-tensor" model) is based on a dynamic law of motion. It allows volumetric deformations and cuttings but has to be applied to a limited number of nodes to compute in real-time.

Our work during my internship is to combine those two approaches into a hybrid model that includes, domain decomposition theory to respect real-time deformations constrains and allows the cutting of large anatomical structures. Finally we present results and conclude with perspectives.

Key-words: Surgery simulation, deformable models, real-time simulation, linear elasticity, domain decomposition.

Optimisation de modèles physiques de Tissus mous pour la simulation chirurgicale.

Résumé: Outre les méthodes permettant de reconstruire des modèles géométriques à partir d'images médicales, il est nécessaire de mettre au point des modèles physiques de tissus mous autorisant une interaction réaliste avec les instruments chirurgicaux. Ainsi, dans la continuité des travaux du projet EPIDAURE sur la simulation de chirurgie hépatique, on se propose de poursuivre l'optimisation des modèles biomecaniques de tissus mous qui doivent être aussi réalistes que possible.

Le simulateur de chirurgie endoscopique développé à EPIDAURE implemente de nombreux modèles physiques. Cependant seuls deux modèles, basés sur la théorie de l'élasticité linéaire et des éléments finis, seront détaillés dans ce rapport de stage. Le premier modèle pré-calcule les déformations et les forces appliquées sur un modèle d'éléments finis et par conséquent autorise le calcul en temps réel de la déformation de maillages ayant beaucoup de sommets. Malheureusement ce maillage ne permet pas de changements de topologie et par conséquent ne permet pas la simulation de découpes lors de la chirurgie. Le second modèle est fondé sur une loi d'évolution dynamique et autorise la simulation de découpes et de déchirements. Nous appelons ce modèle "masses-tenseurs". Avec ce second modèle, il est possible de simuler les déformations et les découpes de structures anatomiques mais uniquement pour des maillages ayant un nombre limité de sommets. Mon travail consiste à faire cohabiter, au sein d'un modèle hybride, le modèle masse-tenseur et le pré-calculé afin de combiner leurs avantages respectifs et ce, en utilisant des techniques reposant sur la décomposition de domaines.

Mots Clés: Simulation de chirurgie, modèles déformables, élasticité linéaire, simulation temps réel, décomposition de domaine.

Remerciements

Je tiens tout d’abord à remercier chaleureusement Nicholas Ayache de m’avoir accueilli au sein du projet EPIDAURE. Merci à Hervé Delingette pour sa patience, sa disponibilité et de m’avoir fait découvrir le monde de la recherche.

Merci aussi à Clément Forest et à Maxime Sermesant pour leur disponibilité et de m’avoir aiguillé au sein des nombreuses classes et fonctions du simulateur.

Je remercie Jean-Didier Lemarechal d’avoir partagé son bureau et pour ses scripts d’installation, Isabelle Strobant pour son sourire et son efficacité, Olivier Clatz d’avoir partagé avec moi ma condition de stagiaire. Un grand merci, à tous les chercheurs et thésards du projet pour leur sympathie et pour l’aide qu’ils ne m’ont jamais refusée.

Enfin, merci à Laurence, pour sa présence et les corrections qu’elle a apportées à ce rapport.

Table des matières

Introduction

Outre les méthodes permettant de reconstruire des modèles géométriques à partir d'images médicales, il est nécessaire de mettre au point des modèles physiques de tissus mous autorisant une interaction réaliste avec les instruments chirurgicaux. Les deux premières modélisations physiques de corps déformables que nous utiliserons se basent, pour des raisons de simplicité de calcul, sur une loi élastique linéaire. On peut considérer la déformation comme linéaire si le déplacement imposé est petit (chaque déplacement doit être inférieur à 10 % de la taille totale du maillage). Le premier modèle correspond à un **modèle quasi statique pré-calculé** très rapide et permettant le temps réel mais étant limité au niveau de l'interaction avec le chirurgien. En effet, la topologie de ce modèle ne peut changer au cours de la simulation. Le chirurgien ne peut donc pas effectuer de découpes sur l'organe modélisé. Le deuxième modèle correspond à un modèle dynamique, le **modèle masse-tenseur**. Ce système offre la possibilité d'autoriser les découpes mais par contre il souffre d'une lourdeur de calcul. Il serait donc intéressant de pouvoir créer une troisième solution correspondant à un **modèle hybride** où cohabitent les deux modèles précédents. Ainsi, grâce à la décomposition du domaine de déformation, nous pourrons appliquer un des deux modèles à chaque sous-domaine de notre organe simulé. Ceci nous permettra de tirer partie de leurs avantages respectifs.

0.1 Le stage

Ce stage se déroule du 2 avril 2002 au 30 septembre 2002 à l'INRIA (Institut National de Recherche en Informatique et Automatique), au sein du projet EPIDAURE. Il constitue la conclusion de ma troisième année d'école d'ingénieur à EURECOM et de mon année de DEA à l'université de Nice Sophia-Antipolis dans le cadre du DEA STIC Image-Vision, dirigé par Monsieur Michel Barlaud. Ce stage est dirigé par Nicholas Ayache, directeur du projet EPIDAURE, et par Hervé Delingette, chercheur du projet EPIDAURE.

0.1.1 Le projet EPIDAURE

L'objectif du projet EPIDAURE est de développer des outils de traitements d'images médicales, avec de nombreuses applications : construction d'atlas anatomiques, simulations d'opérations de chirurgie, recalage d'images volumiques, aide au diagnostic préopératoire, réalité virtuelle et augmentée. Le projet EPIDAURE regroupe cinq chercheurs, le Dr. Nicholas Ayache, son directeur, Xavier Pennec, Hervé Delingette, Grégoire Malandain et Miguel Gonzalez, ainsi qu'Eric Bardinet, ingénieur expert, Jean-Didier Lemarechal, ingénieur, Isabelle Strobant, assistante du projet, une dizaine de thésards, de nombreux collaborateurs, appartenant aussi bien au milieu médical qu'à celui de la recherche scientifique, et bien sûr des stagiaires.

0.1.2 Le sujet de stage

Optimisation de modèles physiques de tissus mous pour la simulation chirurgicale.

Dans la continuitéé des travaux du projet EPIDAURE sur la simulation de chirurgie hépatique, on se propose de poursuivre l'optimisation des modèles biomécaniques de tissus mous qui doivent être aussi réalistes que possible. Plus précisément, et comme décrit dans l'introduction, le simulateur de chirurgie endoscopique dispose de nombreux modèles physiques dont les plus intéressants sont :

- le modèle Masse-tenseur qui permet de découper le foie mais qui est très lourd en terme de calcul,
- le modèle pré-calculé qui est très rapide mais qui ne permet pas de faire des découpes

Mon travail consiste à faire cohabiter au sein d'un modèle hybride le modèle masse-tenseur et le pré-calculé en utilisant des techniques basées sur la décomposition de domaines,. Et ceci de la façon la plus dynamique possible.

0.2 Plan

Nous allons, tout d'abord, décrire brièvement dans une première partie le travail déjà réalisé au sein du projet EPIDAURE par les précédents chercheurs et thésards : dans le premier chapitre, les théories des lois élastiques et de la méthode des éléments finis appliquées au simulateur. Puis, je détaillerai dans les deux chapitres suivants, les deux modèles de base de tissus mous ([2] et [1]). Enfin, dans la deuxième partie j'expliciterai mon travail réalisé pour l'instant au cours de mon stage : ainsi dans le chapitre 3.7,et en me basant sur la théorie de la décomposition de domaine, je définirai un algorithme faisant cohabiter les deux modèles mécaniques de tissus mous. Les chapitres suivants seront consacrés à la présentation des résultats que nous avons obtenus pour l'instant et à une discussion les concernant.

0.3 Le simulateur de chirurgie hépatique

0.3.1 Schéma général de fonctionnement

Un certain nombre de simulateurs de chirurgie ont vu le jour ces dernières années. Si les buts recherchés et les applications peuvent être très différents, l'architecture et le schéma général de fonctionnement de ces simulateurs sont toujours à peu près les mêmes. La pièce maîtresse est le modèle déformable représentant la géométrie et le comportement biomécanique de l'organe à opérer. La seconde partie du simulateur est constituée d'un moyen d'interaction entre l'utilisateur et le modèle déformable. Cette interaction se fera en général par l'intermédiaire d'une interface mécanique, équipée ou non d'un dispositif à retour d'effort.

Dans le cas de notre simulateur de chirurgie hépatique en laparoscopie, les deux parties principales seront un modèle déformable représentant un foie et une interface à retour d'effort respectant l'ergonomie des instruments utilisés par les chirurgiens. Comme représenté sur la figure 2 la boucle gérant l'interface à retour d'effort envoie la position et récupère les forces qu'elle doit appliquer. Pour obtenir un retour d'effort réaliste, cette boucle doit fonctionner à une fréquence de l'ordre de 1000 Hz. La seconde boucle récupère la position

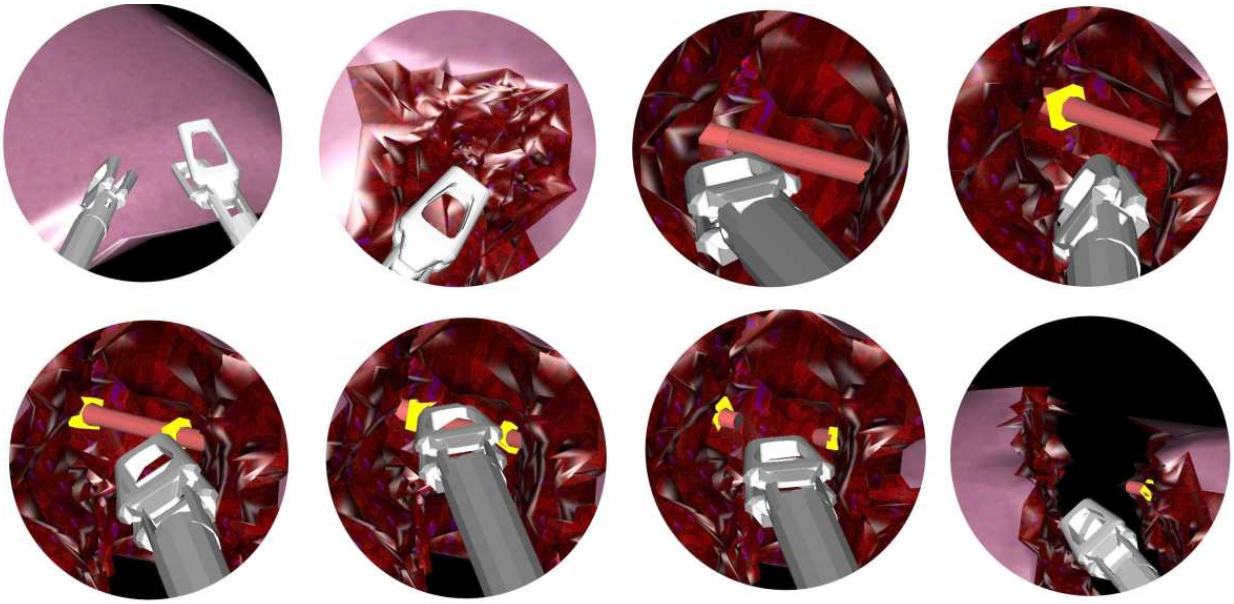


FIG. 1 – *Plusieurs captures d'écran du simulateur de chirurgie laparoscopique sur le foie. (exploration avec les instruments et découpe d'un vaisseau sanguin)*

de l'instrument, détecte les collisions entre l'instrument et le modèle déformable, traduit ces collisions en contraintes qui sont appliquées au modèle déformable afin de modéliser le contact, calcule la déformation et la force exercée sur l'instrument. Cette force est renvoyée au système à retour d'effort. Enfin, cette boucle met à jour l'affichage de la scène à une fréquence de 25 Hz.

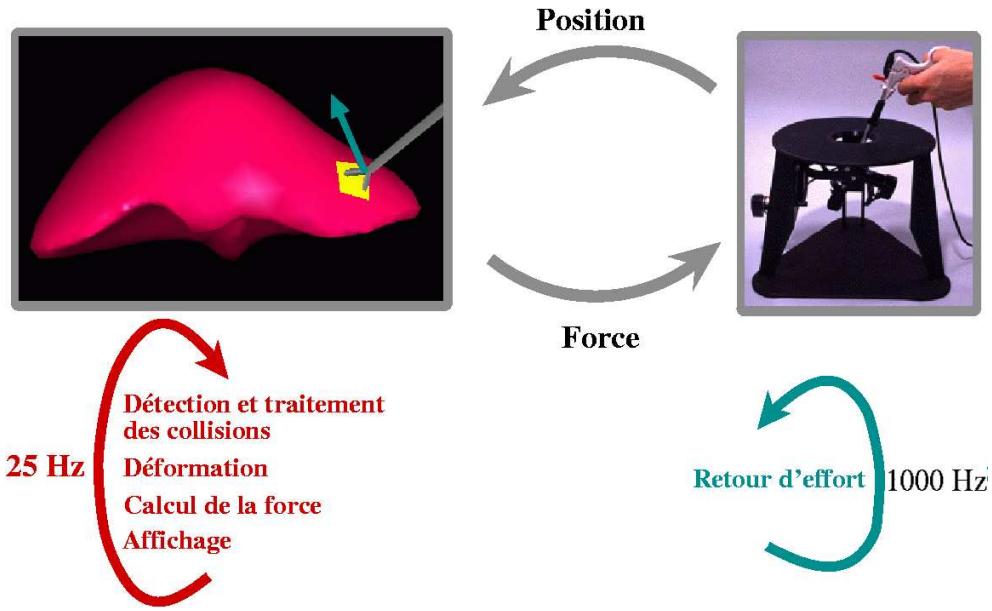


FIG. 2 – *Architecture du simulateur de chirurgie laparoscopique sur le foie.*

Dans le cadre de mon stage je ne m'occuperai que de la partie modèle déformable et plus particulièrement seulement de deux types des modèles implémentés dans le simulateur d'EPIDAURE :

- **le Modèle quasi-static pré-calculé**.
- **le modèle dynamique masse-tenseur**.

0.3.2 Intérêt d'un simulateur en réalité virtuelle

Un simulateur de chirurgie en réalité virtuelle peut être utilisé dans plusieurs contextes différents. Le premier est, bien évidement, la formation et l'entraînement aux gestes. Dans ce cadre, on parlera de simulateurs généraux, dans la mesure où ils sont sensés représenter un patient moyen avec une anatomie "d'école". Ils permettront aux chirurgiens de se familiariser avec un environnement nouveau et d'acquérir les réflexes correspondants.

Mais on peut aussi envisager des simulateurs spécifiques, qui intégreront l'anatomie et les caractéristiques d'un patient particulier. Par exemple dans le cas de la chirurgie hépatique, il est envisageable de reconstruire à partir du scanner du patient la géométrie du foie et des organes environnants.

Un des grands avantages d'un simulateur en réalité virtuelle sera de pouvoir provoquer des évènements inattendus à n'importe quel instant. L'élève chirurgien pourra donc être confronté à des difficultés supplémentaires qui ne sont rencontrées que très rarement lors des opérations réelles, mais auxquelles il doit être préparé. Ces situations particulières peuvent être provoquées par un second utilisateur, ou bien simplement de manière aléatoire.

Enfin, un simulateur de chirurgie peut aussi permettre d'évaluer les qualités et les défauts d'un geste à partir de critères précis. On pourra, par exemple, surveiller la vitesse de déplacements des instruments, la précision de certains gestes ou l'intensité des contraintes appliquées sur l'organe virtuel.

0.4 A propos du foie

Le foie est un organe particulièrement important, qui joue de nombreux rôles essentiels au bon fonctionnement de l'organisme : il produit la bile (qui élimine les substances toxiques et facilite la digestion), ainsi que de nombreuse protéines essentielles; il assainit le sang, régule le taux de glucose, le taux de cholestérol, l'apport en vitamines et en minéraux; il équilibre la circulation de nombreuses hormones ... C'est l'organe le plus volumineux du corps humain. Il mesure typiquement 28 cm dans le sens transversal, 16 cm de haut et 8 cm d'épaisseur. Il est d'une consistance plutôt ferme, fragile mais très malléable : sa variabilité est très importante. Sa forme dépend des ligaments qui le maintiennent et des organes qui se trouvent a coté de lui : les poumons, le cœur, l'estomac, le pancréas, la rate, la vésicule biliaire, les intestins, le diaphragme, les cotes... Par ailleurs, certaines pathologies du foie peuvent influer fortement sur la taille et la forme de celui-ci.

Le foie est doté d'un étonnant pouvoir de regéneration, lui permettant de recouvrer la totalité de sa masse quatre mois après ablation des trois quart de l'organe. Ce pouvoir de régénération dépasse celui des cancers les plus graves. L'ablation, même massive, d'une partie du foie est donc une solution particulièrement efficace pour traiter certaines pathologie hépatique graves. On considère souvent que le foie comprend plusieurs parties, dénommées segments. Ce genre de description est fondée sur la vascularisation du foie. Si une tumeur se

développe dans le foie, elle va utiliser la vascularisation porte pour se propager; il est donc préférable d'ôter tout le segment correspondant, et il n'est pas nécessaire, si l'on s'y prend a temps, de toucher aux segments non concernés. Il existe plusieurs segmentation du foie, mais celle qui est la plus utilisée est celle de Couinaud (voir figure 3).

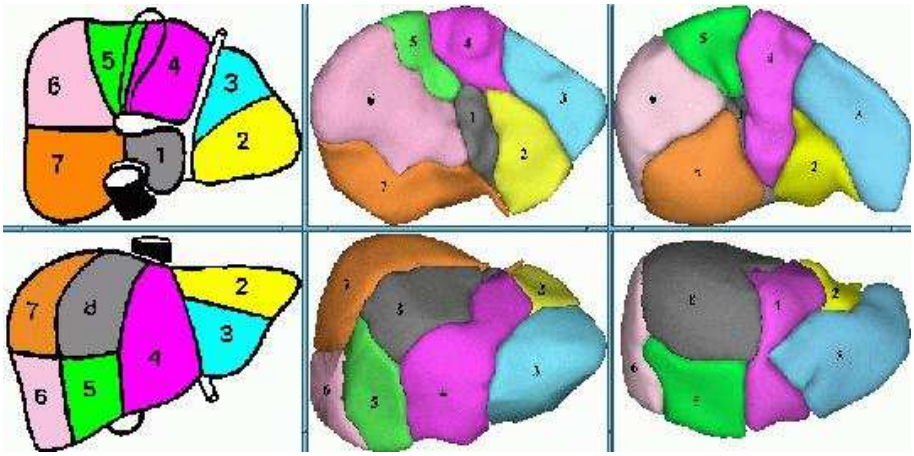


FIG. 3 – *Segmentation de Couinaud. De gauche à droite: schéma, puis deux exemples de modèles 3D de foies réels. En haut vue du dessous; en bas, vue de face*

Première Partie : Les travaux précédents

Chapitre 1

Construction d’un modèle déformable et théorie

La brique principal du simulateur de chirurgie correspond aux modèles physiques déformables qui permettent à notre organe de se déformer de façon réaliste. Il est donc très important.

1.1 Construction d’un modèle déformable

La construction d’un modèle déformable se fait en plusieurs étapes:

– Il faut, tout d’abord, un modèle géométrique qui représente l’organe auquel on s’intéresse. Nous travaillons avec des maillages tétraédriques, particulièrement bien adaptés pour modéliser des objets à la géométrie complexe.

– Il faut ensuite choisir le comportement biomécanique que l’on désire donner à l’organe en question. Afin de garantir le réalisme physique de ce comportement, nous utilisons des lois de déformation issues de la mécanique des milieux continus. Dès lors que le temps de calcul est un facteur important, l’élasticité linéaire semble le modèle le plus approprié. En effet, sa linéarité lui confère une bonne efficacité et permet de lui appliquer plusieurs méthodes d’accélération et d’optimisation, dont certaines seront présentées dans ce chapitre. De plus, la relative simplicité des équations qu’il engendre permet une interprétation "intuitive" des différents comportements, ce qui permet d’y apporter de nombreuses améliorations.

– L’étape suivante consiste à discrétiser le problème élastique obtenu afin de permettre le calcul d’une solution numérique. Lors de cette discrétisation, il faut prévoir la possibilité d’appliquer sur le modèle un ensemble de contraintes aux bords. C’est par l’intermédiaire de ces contraintes que l’utilisateur pourra interagir avec l’organe virtuel.

– Enfin, il faut résoudre le problème, c’est à dire calculer la déformation subie par l’organe et éventuellement en déduire d’autres informations utiles, comme par exemple les contraintes dans les zones de contact entre l’organe et le monde extérieur.

1.2 Théorie

1.2.1 Énergie potentielle élastique : loi de *Hooke*

En se basant sur la *loi de Hooke*:

$$W^l = \frac{\lambda}{2}\,(trE_l)^2 + \mu\, trE_l{}^2. \tag{1.1}$$

Celle ci définie l'énergie élastique d'un objet déformable $\mathbf{W}^l$ comme une fonction quadratique du vecteur déplacement $\mathbf{U}(x,y,z)$ avec $\mathbf{E}^l$ le tenseur de déformation :

$$E_l = \frac{1}{2}\left(\nabla \mathbf{U}^t + \nabla \mathbf{U}\right) \tag{1.2}$$

Pour information, si l'on applique les principes du calcul variationnel, on obtient l'équation aux dérivées partielles (EDP) gouvernant les déformations linéaires élastiques dans le cas statique (se referer à [2] pour plus d'information):

$$\mu\,\Delta\mathbf{U} + (\lambda+\mu)\,\nabla\left(div\,\mathbf{U}\right) + \mathbf{f} = \mathbf{0}, \tag{1.3}$$

où **f** est le champ de forces volumiques externes appliquées sur le domaine. On vérifie bien que cette équation est linéaire par rapport aux dérivées partielles du champ de déplacements.

1.2.2 L'élasticité linéaire en éléments finis

La méthode des éléments finis consiste à subdiviser le domaine sur lequel on travaille pour ensuite approximer, sur chaque sous domaine, le problème que l'on se pose en l'exprimant comme une interpolation de valeurs calculées en certains points. Dans notre cas, chaque sous domaine est un tétraèdre dans lequel le problème est approximé par l'interpolation linéaire de ses valeurs sur les quatre sommets.

Nous considérons donc un maillage tétraédrique de l'organe qui nous intéresse, sur lequel nous allons discrétiser l'énergie élastique de déformation exprimée par l'équation 1.1. La première étape consiste à exprimer la valeur du champ de déplacements **U** à l'intérieur de chaque tétraèdre en fonction des déplacements de ses sommets.

$$\boxed{\mathbf{U}(\mathbf{X}) = \sum_{j=0}^{3} \Lambda_j(\mathbf{X})\,\mathbf{U}_j} \tag{1.4}$$

où
$$\begin{cases} \Lambda_j(\mathbf{X}) = \alpha_j \,.\, \mathbf{X} + \beta_j,\ j = 0..3, \\ \text{sont les fonctions d'interpolation linéaires, et} \\ \\ \mathbf{U}_j = \mathbf{U}(\mathbf{P}_j),\ j = 0..3, \\ \text{sont les déplacements des sommets.} \end{cases}$$

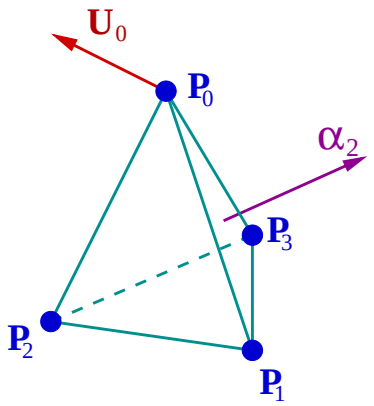


FIG. 1.1 – *Élément fini de type P1*

Nous pouvons ainsi exprimer le gradient du champ de déplacement :

$$\nabla \mathbf{U} = \sum_{j=0}^{3} \mathbf{U}_j \, \nabla\big(\alpha_j \,.\, \mathbf{X} + \beta_j\big) = \sum_{j=0}^{3} \mathbf{U}_j \, \alpha_j^t$$

$$\Longrightarrow \boxed{\nabla \mathbf{U} = \sum_{j=0}^{3} \mathbf{U}_j \otimes \alpha_j} \tag{1.5}$$

où $\otimes$ est le produit tensoriel de deux vecteurs.

On remarque que ce gradient est constant à l'intérieur du tétraèdre, et qu'il sera donc représenté sur tout le maillage par une fonction constante par morceaux.

On va maintenant pouvoir exprimer le tenseur de déformation E_l, les deux termes $(tr\ E_l)^2$ et $tr\ {E_l}^2$, pour finalement obtenir l'expression de l'énergie élastique de déformation à l'intérieur d'un tétraèdre :

$$E_l = \frac{1}{2}\left(\nabla \mathbf{U}^t + \nabla \mathbf{U}\right) = \frac{1}{2}\sum_{j=0}^{3}\left(\mathbf{U}_j \otimes \alpha_j + \alpha_j \otimes \mathbf{U}_j\right) \tag{1.6}$$

$$\big(tr\ E_l\big)^2 = \left(\sum_{j=0}^{3} \mathbf{U}_j \,.\, \alpha_j\right)^2 = \sum_{j,k=0}^{3} \mathbf{U}_j^t\big[\alpha_j \otimes \alpha_k\big]\mathbf{U}_k \tag{1.7}$$

$$\begin{aligned} tr\ {E_l}^2 = \frac{1}{4} tr\Bigg(& \sum_{j,k=0}^{3} (\mathbf{U}_j \otimes \alpha_k)(\mathbf{U}_k \,.\, \alpha_j) + (\alpha_j \otimes \mathbf{U}_k)(\mathbf{U}_j \,.\, \alpha_k) \\ & + (\mathbf{U}_j \otimes \mathbf{U}_k)(\alpha_j \,.\, \alpha_k) + (\mathbf{U}_j \,.\, \mathbf{U}_k)(\alpha_j \otimes \alpha_k)\Bigg) \\ = \frac{1}{2} & \sum_{j,k=0}^{3} \mathbf{U}_j^t\big[(\alpha_k \otimes \alpha_j) + (\alpha_j \,.\, \alpha_k)\, Id\big]\mathbf{U}_k \end{aligned} \tag{1.8}$$

$$\boxed{W^l(\mathcal{T}) = \frac{1}{2}\sum_{j,k=0}^{3} \mathbf{U}_j^t\big[\mathcal{B}_{\mathcal{T}}^{jk}\big]\mathbf{U}_k \quad \text{avec} \quad \mathcal{B}_{\mathcal{T}}^{jk} = \lambda\big(\alpha_j \otimes \alpha_k\big) + \mu\Big[(\alpha_k \otimes \alpha_j) + (\alpha_j \,.\, \alpha_k)\, Id\Big]} \tag{1.9}$$

L'ensemble des matrices (3x3) $\left\{\mathcal{B}_{\mathcal{T}}^{jk}\ ;\ j,k = 0..3\right\}$ constitue l'ensemble des tenseurs locaux de rigidité du tétraèdre $\mathcal{T}$. Ils représentent les caractéristiques locales du matériau élastique. Si $j = k$, on parle du tenseur local de rigidité du sommet $\mathbf{P}_j$, sinon du tenseur local de rigidité de l'arête $(\mathbf{P}_j, \mathbf{P}_k)$. On montre facilement que $[\mathcal{B}_{\mathcal{T}}^{kj}] = [\mathcal{B}_{\mathcal{T}}^{jk}]^t$. De ce fait, seulement 10 tenseurs sont nécessaires pour décrire le comportement du matériau à l'intérieur de ce tétraèdre.

En dérivant cette énergie de déformation par rapport à la position d'un des sommets $\mathbf{P}_p$, on obtient la force interne $\mathbf{F}_p^l(\mathcal{T})$ exercée sur ce sommet par l'élément de matière élastique que représente le tétraèdre $\mathcal{T}$. On se rend alors compte que cette force peut se décomposer en deux parties. La première partie est la force créée par le déplacement du sommet lui-même. Elle a tendance à rappeler le sommet vers sa position de repos. La deuxième force est engendrée par le déplacement des autres sommets du tétraèdre. Elle traduit l'influence que les sommets exercent sur leurs voisins. C'est donc elle qui va permettre la propagation de la déformation à l'intérieur du matériau. La donnée des forces élastiques exercées en chaque sommet permet de caractériser le comportement élastique linéaire de ce tétraèdre :

$$\mathbf{F}_p^l(\mathcal{T}) = \sum_{j=0}^{3} \left[\mathcal{B}_{\mathcal{T}}^{pj}\right] \mathbf{U}_j$$

$$\Longleftrightarrow \boxed{\mathbf{F}_p^l(\mathcal{T}) = \left[\mathcal{B}_{\mathcal{T}}^{pp}\right] \mathbf{U}_p + \sum_{\substack{j=0 \\ j\neq p}}^{3} \left[\mathcal{B}_{\mathcal{T}}^{pj}\right] \mathbf{U}_j} \quad (1.10)$$

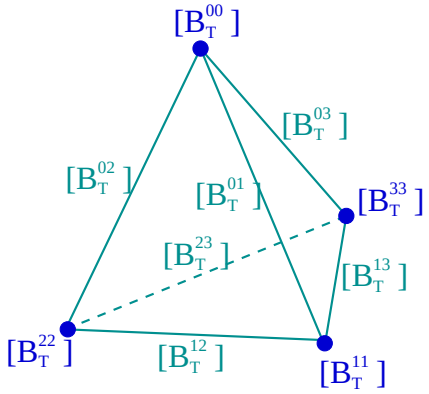


FIG. 1.2 – *Répartition des tenseurs de rigidité sur les sommets et sur les arêtes du tétraèdre*

Si on considère maintenant non plus un seul tétraèdre mais le maillage tétraédrique du domaine tout entier, on doit accumuler les contributions de chaque élément. La force interne globale $\mathbf{F}_p^l$ exercée sur un sommet se calcule alors comme la somme des forces internes calculées sur chaque tétraèdre adjacent à ce sommet. En posant cette somme, on s'aperçoit que l'on peut mettre en facteur le déplacement du sommet et celui de chacun de ses voisins pour finalement obtenir l'équation de la force interne sous la forme :

$$\begin{aligned}\mathbf{F}_p^l &= \sum_{\mathcal{T} \in \nu(\mathbf{P}_p)} \mathbf{F}_p^l(\mathcal{T}) = \sum_{\mathcal{T} \in \nu(\mathbf{P}_p)} \left(\left[\mathcal{B}_{\mathcal{T}}^{pp}\right] \mathbf{U}_p + \sum_{\substack{j=0 \\ j\neq p}}^{3} \left[\mathcal{B}_{\mathcal{T}}^{pj}\right] \mathbf{U}_j \right) \\ &= \underbrace{\left(\sum_{\mathcal{T} \in \nu(\mathbf{P}_p)} \left[\mathcal{B}_{\mathcal{T}}^{pp}\right] \right)}_{\left[\mathcal{B}^{pp}\right]} \mathbf{U}_p + \sum_{\mathcal{A} \in \nu(\mathbf{P}_p)} \underbrace{\left(\sum_{\mathcal{T} \in \nu(\mathbf{P}_p, \mathbf{P}_j)} \left[\mathcal{B}_{\mathcal{T}}^{pj}\right] \right)}_{\left[\mathcal{B}^{pj}\right]} \mathbf{U}_j \end{aligned} \quad (1.11)$$

Avec les notations suivantes pour les sommes :

- $\mathcal{T} \in \nu(\mathbf{P}_p)$: *"somme sur tous les tétraèdres appartenant au voisinage du sommet* $\mathbf{P}_p$*"*,
- $\mathcal{A} \in \nu(\mathbf{P}_p)$: *"somme sur toutes les arêtes appartenant au voisinage du sommet* $\mathbf{P}_p$*"*, et
- $\mathcal{T} \in \nu(\mathbf{P}_p, \mathbf{P}_j)$: *"somme sur tous les tétraèdres appartenant au voisinage de l'arête* $(\mathbf{P}_p, \mathbf{P}_j)$*"*.

Lors de la construction de la structure de données liée au maillage, nous allons calculer, pour chaque tétraèdre, les tenseurs de rigidité locaux et accumuler leurs contributions sur les sommets et les arêtes. Sur la figure 1.3, le tenseur de rigidité du sommet identifié par un point ou de l'arête centrale est la somme des contributions de tous les tétraèdres adjacents. On obtient ainsi des tenseurs de rigidité **globaux** pour chaque sommet ($\{\mathcal{B}^{pp}\}$) et pour chaque arête ($\{\mathcal{B}^{pj},\ p \neq j\}$) du maillage.

La force interne globale exercée sur chaque nœud du maillage s'exprime donc en fonction du déplacement de ce nœud et des déplacements de ses voisins par la formule suivante :

$$\boxed{\mathbf{F}_p^l = \left[\mathcal{B}^{pp}\right] \mathbf{U}_p + \sum_{\mathcal{A} \in \nu(\mathbf{P}_p)} \left[\mathcal{B}^{pj}\right] \mathbf{U}_j} \quad (1.12)$$

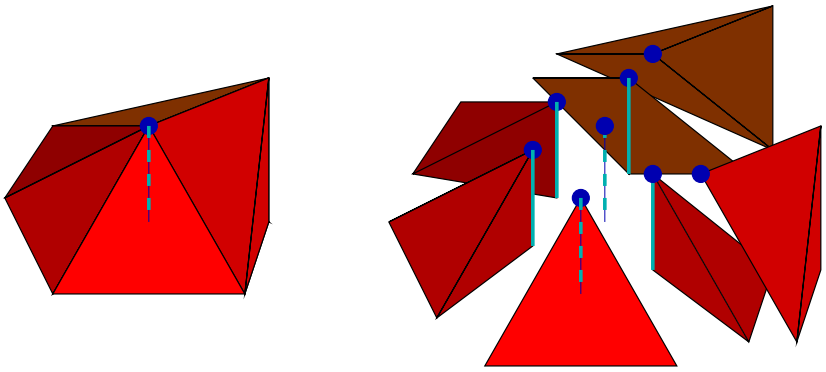


FIG. 1.3 – *Accumulation des tenseurs de rigidité sur les sommets et les arêtes du maillage*

1.3 Conclusion

Soit N_s le nombre de sommets du maillage. Nous disposons alors d'un ensemble de N_s équations du type 1.12 exprimant les forces élastiques internes exercées sur chacun des nœuds du maillage en fonction du champ de déplacements local. Chaque équation étant vectorielle, nous avons affaire à un système de $3 * N_s$ équations à $3 * N_s$ inconnues.

Avant de pouvoir résoudre ce problème, nous allons devoir modifier certaines équations afin de prendre en compte les conditions aux bords et les contraintes que nous voulons appliquer sur notre modèle :

- certains noeuds peuvent être fixés, ce qui correspond pour les organes du corps humain, aux zones en contact avec des structures voisines ou directement attachées sur les os.
- On peut aussi imposer un déplacement non nul, ce qui revient à pré-contraindre le modèle.
- Enfin, on peut appliquer des forces sur les sommets de la surface (forces de pression ou de contact) ou sur les sommets internes comme par exemple lorsque l'on veut modéliser l'action de la gravité.

On pourra alors résoudre le problème global et en déduire la déformation du modèle, c'est-à-dire calculer l'ensemble des déplacements des noeuds du maillage.

Dans les deux chapitres suivants, nous allons présenter les deux méthodes de résolution que nous utilisons. La première met en œuvre une résolution implicite du problème, accélérée par l'utilisation de pré-calculs, alors que la seconde utilise un schéma explicite dynamique.

Chapitre 2

L'élasticité linéaire : Modèle pré-calculé

2.1 Introduction

Ce chapitre est consacré à la première méthode. Cette résolution implicite des équations du précédent chapitre, est une méthode quasi-statique couplée à l'utilisation de pré-calculs. Il est classique de présenter le problème de l'élasticité linéaire discrétisée par la méthode des éléments finis sous la forme d'un unique système d'équations linéaires qui s'écrit sous la forme $\mathbf{F} = \mathbf{K}.\mathbf{U}$. L'ensemble $\mathbf{U}$ des déplacements des sommets du maillage sera alors obtenu en résolvant le système par une méthode itérative du type *gradient conjugué*. Malheureusement, le temps de calcul de ce type de résolution rend son utilisation impossible dans le cadre d'une application temps-réel. Par exemple, avec un maillage comprenant 1313 sommets, ce qui donne un système linéaire (3939 x 3939), il faut 140 itérations de gradient conjugué [1] pré-conditionné (factorisation de Cholesky ou LU incomplètes) pour atteindre une précision de 10^{-6} *m*, ce qui prend environ 9 secondes sur un PC Pentium II (450 MHz) sous Linux.

Pour contourner ce problème de temps de calculs, nous utilisons une méthode dérivée de celle développée par Cotin et al. [1] et améliorée par Picinbono [2]. Cette méthode est basée sur le fait que, grâce à la linéarité du modèle, la déformation obtenue en appliquant la somme de deux contraintes est bien la somme des déformations obtenues en appliquant chaque contrainte indépendamment (figure 2.1).

Donc, à partir du moment où l'on a calculé les déformations engendrées par l'application de certaines contraintes, on est capable de retrouver directement les déformations correspondant à l'application d'une combinaison linéaire de ces contraintes. L'idée consiste alors à pré-calculer une base de déformations élémentaires et à se servir de cette base pour exprimer n'importe quelle déformation du modèle.

2.2 Construction de la base de déformations (avant la simulation)

La première chose à faire est d'assembler la matrice de rigidité globale du système à partir des matrices de rigidité élémentaires des sommets et des arêtes (les $\mathcal{B}^{pp}$ et $\mathcal{B}^{pj}$ de l'équation 1.12). Cette matrice K est de dimension ($3N_s$ x $3N_s$) (où N_s est le nombre de sommets du maillage). Une fois cette matrice assemblée, le

1. Nous utilisons la librairie d'algèbre linéaire *"Matrix Template Library"*: http://www.lsc.nd.edu/research/mtl/ et http://www.lsc.nd.edu/research/itl/

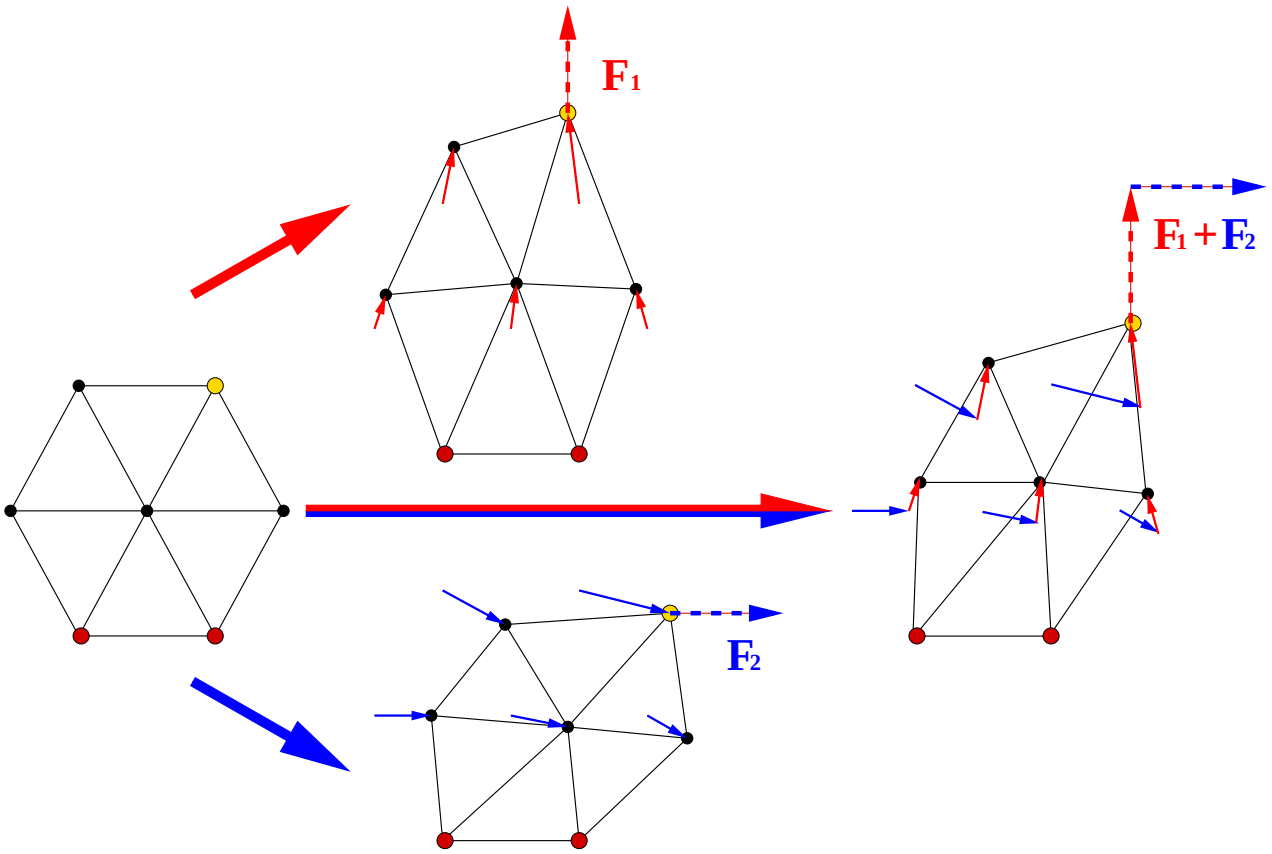


FIG. 2.1 – *À gauche se trouve le maillage au repos. Les deux sommets du bas sont fixés, et on applique les forces sur le sommet en haut à droite. Les déplacements obtenus en appliquant la somme des deux forces* $\mathbf{F}_1 + \mathbf{F}_2$ *(à droite) sont égaux à la somme des déplacements obtenus en appliquant indépendamment la force* $\mathbf{F}_1$ *(en haut) et la force* $\mathbf{F}_2$ *(en bas).*

problème peut s'écrire sous la forme $[K]\,\mathbf{U} = \mathbf{F}$, ou encore :

$$\begin{bmatrix} \mathcal{B}^{0,0} & \dots & \mathcal{B}^{0,p} & \dots & \mathcal{B}^{0,N_s-1} \\ \vdots & \ddots & \vdots & & \vdots \\ \mathcal{B}^{p,0} & \dots & \mathcal{B}^{p,p} & \dots & \mathcal{B}^{p,N_s-1} \\ \vdots & & \vdots & \ddots & \vdots \\ \mathcal{B}^{N_s-1,0} & \dots & \mathcal{B}^{N_s-1,p} & \dots & \mathcal{B}^{N_s-1,N_s-1} \end{bmatrix} \begin{bmatrix} \mathbf{U}_0 \\ \vdots \\ \mathbf{U}_p \\ \vdots \\ \mathbf{U}_{N_s-1} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_0 \\ \vdots \\ \mathbf{F}_p \\ \vdots \\ \mathbf{F}_{N_s-1} \end{bmatrix} \tag{2.1}$$

Il faut remarquer que la matrice K est creuse puisque la matrice élémentaire $\mathcal{B}^{pj}$ n'est non nulle que si il existe une arête reliant les sommets $\mathbf{P}_p$ et $\mathbf{P}_j$. Or dans un maillage tétraédrique régulier, chaque sommet n'est relié qu' à une dizaine d'arêtes.Par exemple, voici la matrice de rigidité pour un solide de 15 nœuds sur la figure 2.2

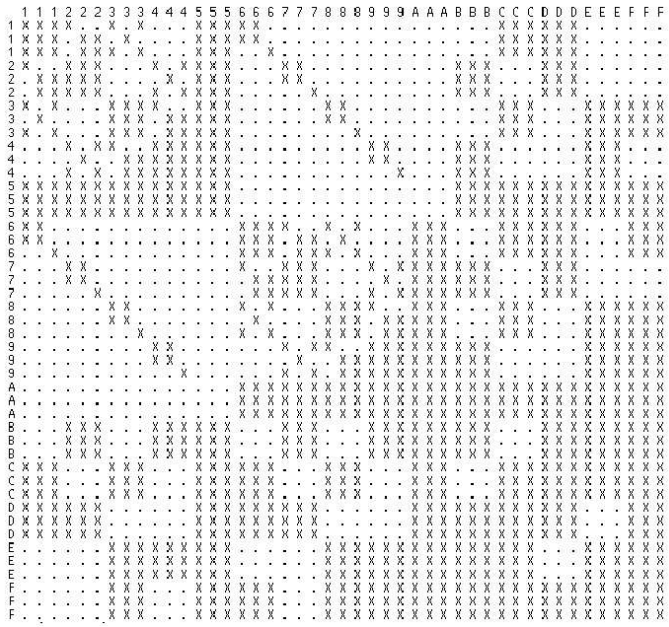


FIG. 2.2 – *Matrice de rigidité d'un solide de 15 nœuds. Les 'X' correspondent aux éléments non nuls les '.' aux éléments nuls. En considérant, par exemple, les noeuds 3 et 5, les valeurs à l'intersection des 3 lignes correspondantes au nœud 3 et des 3 colonnes correspondantes au noeud 5 contiennent des valeurs non nulles. Il s'en suit qu'une liaison existe entre les noœuds 3 et 5.*

Il faut ensuite définir des conditions aux limites, ce qui revient à modifier le système linéaire 2.1. Ces modifications sont plus ou moins compliquées suivant le cas. Le plus simple est d'imposer des forces $\mathbf{F}_p^*$ sur

certains sommets. Il suffit pour cela de modifier le second membre du système :

$$\begin{bmatrix} \mathcal{B}^{0,0} & \dots & \mathcal{B}^{0,p} & \dots & \mathcal{B}^{0,N_s-1} \\ \vdots & \ddots & \vdots & & \vdots \\ \mathcal{B}^{p,0} & \dots & \mathcal{B}^{p,p} & \dots & \mathcal{B}^{i,N_s-1} \\ \vdots & & \vdots & \ddots & \vdots \\ \mathcal{B}^{N_s-1,0} & \dots & \mathcal{B}^{N_s-1,p} & \dots & \mathcal{B}^{N_s-1,N_s-1} \end{bmatrix} \begin{bmatrix} \mathbf{U}_0 \\ \vdots \\ \mathbf{U}_p \\ \vdots \\ \mathbf{U}_{N_s-1} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_0 \\ \vdots \\ \boxed{\mathbf{F}_p^*} \\ \vdots \\ \mathbf{F}_{N_s-1} \end{bmatrix} \tag{2.2}$$

Mais on peut aussi imposer les déplacements $\mathbf{U}_p^*$ de certains noeuds. Cette fois, il faut modifier la matrice de rigidité. En effet, imposer le déplacement d'un sommet revient à enlever sa contribution dans la déformation, en la redistribuant sous forme de forces dans le second membre :

$$\begin{bmatrix} \mathcal{B}^{0,0} & \dots & \begin{matrix}0&0&0\\0&0&0\\0&0&0\end{matrix} & \dots & \mathcal{B}^{0,N_s-1} \\ \vdots & \ddots & \vdots & & \vdots \\ \begin{matrix}0&0&0\\0&0&0\\0&0&0\end{matrix} & \cdots & \begin{matrix}1&0&0\\0&1&0\\0&0&1\end{matrix} & \cdots & \begin{matrix}0&0&0\\0&0&0\\0&0&0\end{matrix} \\ \vdots & & \vdots & \ddots & \vdots \\ \mathcal{B}^{N_s-1,0} & \dots & \begin{matrix}0&0&0\\0&0&0\\0&0&0\end{matrix} & \dots & \mathcal{B}^{N_s-1,N_s-1} \end{bmatrix} \begin{bmatrix} \mathbf{U}_0 \\ \vdots \\ \mathbf{U}_p \\ \vdots \\ \mathbf{U}_{N_s-1} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_0 \boxed{-\mathcal{B}^{0,p}\,\mathbf{U}_p^*} \\ \vdots \\ \boxed{\mathbf{U}_p^*} \\ \vdots \\ \mathbf{F}_{N_s-1} \boxed{-\mathcal{B}^{N_s-1,p}\,\mathbf{U}_p^*} \end{bmatrix} \tag{2.3}$$

Il est intéressant de remarquer que dans le cas où le déplacement imposé est nul, c'est-à-dire lorsque l'on veut fixer un point, il suffit de retirer la ligne et la colonne qui lui correspondent dans le système linéaire.

Si cette méthode est souvent utilisée en pratique, elle n'est pas correcte d'un point de vue mathématique. En effet, en modifiant ainsi la matrice de rigidité et le second membre, on change la dimension de l'espace vectoriel dans lequel on résout le système d'équations, et on ne garantit plus que la solution finale corresponde à un minimum global.

Une fois les conditions aux limites imposées, on peut procéder à la construction de la base de déformation. Pour cela on considère un sommet $\mathbf{P}_l$ du maillage. On lui applique une force élémentaire $\mathbf{F}_x^e$ dans la direction x (en modifiant le second membre du système, comme nous venons de le voir) et on résout le système linéaire. On obtient donc les déplacements de tous les noeuds du maillage engendrés par l'application de cette force élémentaire. Alors, pour chaque sommet du maillage $\mathbf{P}_m$, on stocke ce déplacement $\mathbf{U}_m^x = [u_m^x \; v_m^x \; w_m^x]^t$ dans la première colonne du tenseur T^{lm}. On réitère l'opération avec des forces élémentaires dans les directions y et z, ce qui permet de trouver la deuxième et la troisième colonne du tenseur T^{lm}. Une fois que l'on a fait cette opération pour tous les sommets du maillage, on sauvegarde dans un fichier l'ensemble des ${N_s}^2$ tenseurs T^{lm} représentant les déplacements subis par le sommet $\mathbf{P}_m$ lorsque le sommet $\mathbf{P}_l$ est soumis à des forces élémentaires dans les trois directions de l'espace :

$$\begin{matrix} \mathbf{F}_x^e & \mathbf{F}_y^e & \mathbf{F}_x^e \\ \downarrow & \downarrow & \downarrow \end{matrix} \tag{2.4}$$

$$T^{lm} = \begin{bmatrix} u_m^x & u_m^y & u_m^z \\ v_m^x & v_m^y & v_m^z \\ w_m^x & w_m^y & w_m^z \end{bmatrix} \tag{2.5}$$

2.3 Utilisation de la base de déformations (pendant la simulation)

L'utilisation de la base de déformations met en œuvre des algorithmes différents suivant les conditions aux bords que l'on veut appliquer au modèle. Nous allons donc envisager les différents cas, en allant du plus simple au plus compliqué :

- **Application d'une force en un seul point** :
  C'est le cas le plus simple. Supposons que l'on veuille appliquer une force $\mathbf{F}_m$ sur le sommet $\mathbf{P}_m$. On commence par exprimer la force $\mathbf{F}_m$ dans la base définie par les forces élémentaires $\mathbf{F}^e$ utilisées lors des pré-calculs :

$$\mathbf{F}_m^{\mathcal{B}_e} = \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} \quad \text{où} \quad \mathbf{F}_m = \begin{bmatrix} \alpha\, \mathbf{F}_x^e \\ \beta\, \mathbf{F}_y^e \\ \gamma\, \mathbf{F}_z^e \end{bmatrix} \tag{2.6}$$

  Remarque : Si on a choisi des forces élémentaires de norme 1, cette première étape est inutile.

  Alors, chaque sommet $\mathbf{P}_l$ du maillage est soumis au déplacement $\mathbf{U}_l$ défini par :

$$\boxed{\mathbf{U}_l = \left[T^{lm}\right]\ \mathbf{F}_m^{\mathcal{B}_e}} \tag{2.7}$$

- **Application de forces en plusieurs points** :
  On applique exactement le même raisonnement que pour le cas précédent, mais avec plusieurs points d'application des forces. Le déplacement de chaque point du maillage est la somme des déplacements engendrés par chacune des forces. Si on nomme S l'ensemble des indices des sommets sur lesquels une force est appliquée, on peut écrire :

$$\boxed{\mathbf{U}_l = \sum_{m \in \mathcal{S}} \left[T^{lm}\right]\ \mathbf{F}_m^{\mathcal{B}_e}} \tag{2.8}$$

- **Application d'un déplacement en un seul point** :
  Pour pouvoir utiliser la base de déformations, il faut se ramener à l'application d'une force. Il va donc falloir calculer la force $\mathbf{F}_m$ qui entraînerait le déplacement $\mathbf{U}_m$ que l'on veut appliquer au sommet $\mathbf{P}_m$. En appliquant la formule 2.7 au déplacement $\mathbf{U}_m$, on peut écrire :

$$\mathbf{U}_m = \left[T^{mm}\right]\ \mathbf{F}_m^{\mathcal{B}_e} \iff \mathbf{F}_m^{\mathcal{B}_e} = \left[T^{mm}\right]^{-1}\ \mathbf{U}_m \tag{2.9}$$

  Il ne reste plus alors qu'à retrouver les déplacements de tous les noeuds $\mathbf{P}_l$ en appliquant la force $\mathbf{F}_m^{\mathcal{B}_e}$ (équation 2.7) :

$$\boxed{\mathbf{U}_l = \left[T^{lm}\right]\ \left[T^{mm}\right]^{-1}\ \mathbf{U}_m} \tag{2.10}$$

- **Application de déplacements en plusieurs points** :
  C'est dans ce cas que les choses se compliquent un peu. En effet, le principe de superposition ne fonctionne plus aussi bien si on raisonne en déplacements. Si on veut imposer des déplacements en plusieurs noeuds, il faut tenir compte des interactions entre les noeuds à déplacer. On peut voir, sur la figure 2.3, le problème qui se pose avec des déplacements imposés sur deux sommets. On voit que si on calcule les

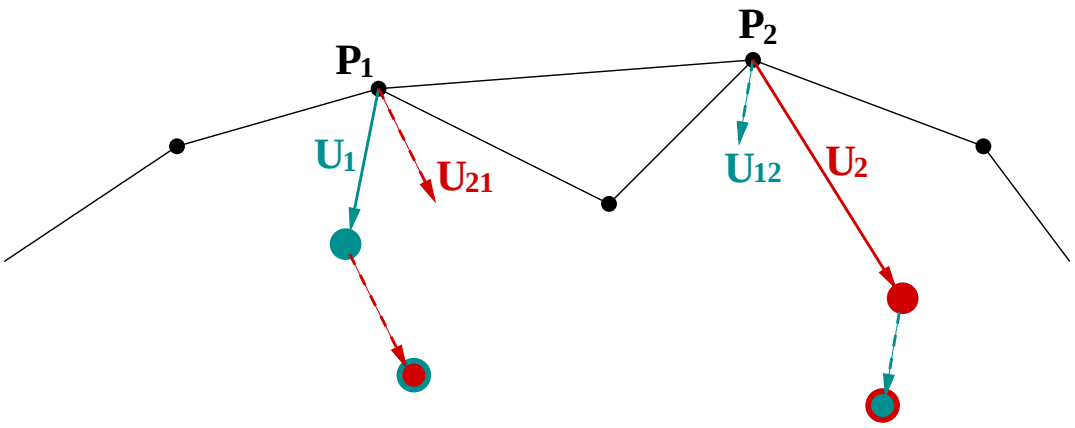


FIG. 2.3 – *Problème posé par l'application de déplacements sur deux sommets simultanément*

forces à appliquer sur les sommets comme dans le cas précédent, le sommet $\mathbf{P}_1$ se déplace de $\mathbf{U}_1$, mais est aussi soumis à l'influence du déplacement du sommet $\mathbf{P}_2$. Cette influence est caractérisée par $\mathbf{U}_{21}$, le déplacement imposé par $\mathbf{P}_2$ sur $\mathbf{P}_1$. Résultat, ces deux sommets se sont trop déplacés. Pour trouver les forces $\mathbf{F}_1^{\mathcal{B}_e}$ et $\mathbf{F}_2^{\mathcal{B}_e}$ qu'il faut réellement appliquer aux deux sommets pour obtenir les déplacements $\mathbf{U}_1$ et $\mathbf{U}_2$, il faut résoudre le système suivant :

$$\begin{cases} \left[T^{11}\right]\mathbf{F}_1^{\mathcal{B}_e} + \underbrace{\left[T^{12}\right]\mathbf{F}_2^{\mathcal{B}_e}}_{\mathbf{U}_{21}} & = \mathbf{U}_1 \\ \underbrace{\left[T^{21}\right]\mathbf{F}_1^{\mathcal{B}_e}}_{\mathbf{U}_{12}} + \left[T^{22}\right]\mathbf{F}_2^{\mathcal{B}_e} & = \mathbf{U}_2 \end{cases} \tag{2.11}$$

De manière plus générale, si on veut appliquer sur n sommets des déplacements $\mathbf{U}_m$ ($m = 0..n-1$), on peut écrire ce système sous forme matricielle :

$$\begin{bmatrix} \left[T^{0,0}\right] & \left[T^{0,1}\right] & \cdots & \left[T^{0,n-1}\right] \\ \left[T^{1,0}\right] & \left[T^{1,1}\right] & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \left[T^{n-1,0}\right] & \cdots & \cdots & \left[T^{n-1,n-1}\right] \end{bmatrix} \begin{bmatrix} \mathbf{F}_0^{\mathcal{B}_e} \\ \vdots \\ \vdots \\ \mathbf{F}_{n-1}^{\mathcal{B}_e} \end{bmatrix} = \begin{bmatrix} \mathbf{U}_0 \\ \vdots \\ \vdots \\ \mathbf{U}_{n-1} \end{bmatrix} \tag{2.12}$$

En résolvant ce système, on trouve les forces $\mathbf{F}_m^{\mathcal{B}_e}$ ($m = 0..n-1$) qu'il faut appliquer sur les sommets $\mathbf{P}_m$.

Le déplacement d'un sommet $\mathbf{P}_l$ du maillage est donc finalement :

$$\boxed{\mathbf{U}_l = \sum_{m=0}^{n-1} \left[T^{lm}\right] \mathbf{F}_m^{\mathcal{B}_e}} \tag{2.13}$$

La figure 2.4 illustre ce cas. On veut appliquer un même déplacement aux trois sommets de la face selectionnée. Le maillage en rendu de surface représente la position au repos et le maillage en fil de fer la position déformée. On voit alors les forces qu'il a effectivement fallu appliquer pour obtenir les déplacements souhaités :

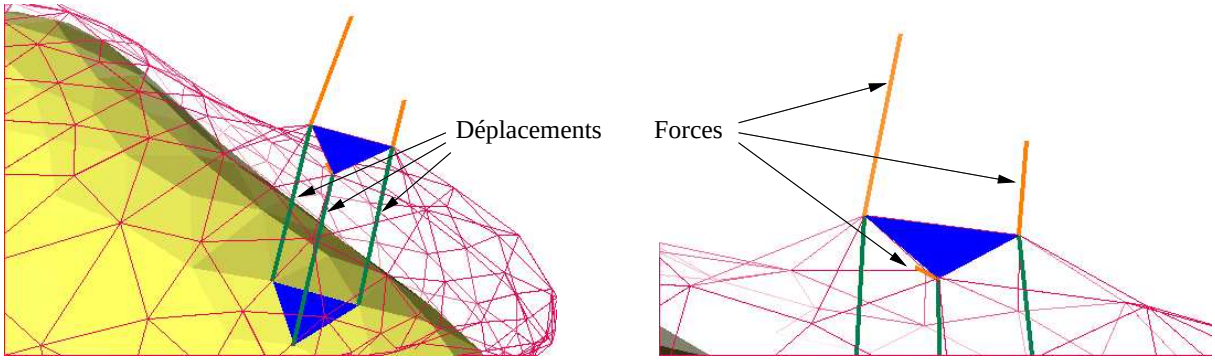


FIG. 2.4 – *Application d'un même déplacement aux trois sommets de la face bleue. On voit en orange les forces qu'il faut appliquer sur ces trois sommets pour obtenir les déplacements souhaités (en vert).*

2.4 Résultats : performances et exemples

Le maillage du foie que nous utilisons contient 1394 sommets et 8347 arêtes pour 6342 tétraèdres. Sa surface est composée de 1224 triangles, ce qui permet d'obtenir une représentation relativement lisse dès lors que l'on utilise des techniques de rendu comme le *Gouraud Shading*.

Ce maillage est fixé par un certain nombre de sommets sur sa face antérieure (figure 2.5), le reste de l'organe étant libre. Dans la mesure où ce modèle déformable subira uniquement des contraintes appliquées

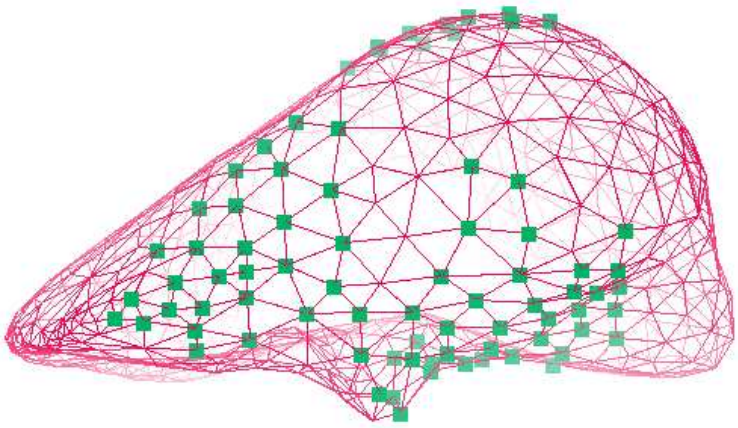


FIG. 2.5 – *Points fixés sur la face antérieure du foie*

sur sa surface (forces de pression ou de contact), nous n'effectuons les pré-calculs que pour les sommets situés sur la surface du maillage, ce qui représente 614 points. Le temps de calcul nécessaire à la construction de la base de déformation va dépendre principalement des coefficients d'élasticité. On peut toutefois donner un ordre d'idée : pour des valeurs $\lambda = 31034\, kg/cm^2$ et $\mu = 3448\, kg/cm^2$ (qui correspondent à $E = 10000\, kg/cm^2$ et $\nu = 0.45$, les pré-calculs demandent un peu plus de 4 heures de calcul. La taille du fichier de stockage des données est de 36 Megabits.

Le tableau 2.1 regroupe les performances atteintes par le modèle déformable pré-calculé lors de l'application de différentes contraintes aux bords. On constate que ce modèle permet largement de garantir un fonctionnement en temps réel (>25 Hz), même dans le cas le moins adapté où l'on veut imposer des déplacements en un grand nombre de nœuds.

Fréquences de simulation (modèle de foie dont la surface est composée de 614 sommets)		Pentium PII 450 MHz	Pentium PIII 600 MHz	SGI Onyx2 IR 195 MHz
Forces sur 1 sommet		3120 Hz	3772 Hz	3846 Hz
Déplacements sur	1 sommet	3081 Hz	3759 Hz	3745 Hz
	5 sommets	419 Hz	561 Hz	510 Hz
	10 sommets	142 Hz	185 Hz	167 Hz
	20 sommets	30 Hz	40 Hz	36 Hz

TAB. 2.1 – *Performances du calcul de déformations pour l'application de différents types de contraintes. Dans le cas de l'application de forces, on ne donne les performances que pour 1 force, car le temps de calcul est linéaire en fonction du nombre de forces appliquées.*

2.5 Bilan

Ce premier modèle déformable permet donc de calculer avec une grande efficacité et une grande précision les déformations d'objets à la géométrie relativement complexe. Mais ces performances sont atteintes au prix d'une restriction assez pénalisante en vue d'une application à la simulation de chirurgie : il est impossible de procéder à des changements de topologie du maillage. En effet, toute modification de la structure du modèle entraîne une remise à jour de la matrice de rigidité globale du système. Or cette remise à jour impose de recommencer les pré-calculs à chaque changement intervenant dans le maillage. Il est donc, en particulier, impossible de simuler une découpe, qui est un des gestes élémentaires effectué par les chirurgiens lors d'interventions chirurgicales.

Pour cette raison, l'équipe d'EPIDAURE ([2]) a proposé un autre modèle déformable basé sur les mêmes théories de l'élasticité linéaire et des éléments finis, et utilisant les mêmes équations pour calculer les forces

exercées en chaque nœud du maillage (équation 1.12), mais dont le schéma de résolution et la structure de données adaptée permettent, à tout instant, de modifier de manière efficace la topologie du maillage.

Chapitre 3

L'élasticité linéaire : Modèle dynamique (Masse-Tenseur)

3.1 Introduction

Contrairement au modèle pré-calculé présenté au paragraphe précédent, le modèle masse-tenseur ne nécessite pas l'assemblage de la matrice globale de rigidité. En effet, toutes les données nécessaires au calcul des forces sont directement stockées dans la structure de données liée au maillage. À partir du maillage tétraédrique de l'objet que l'on veut simuler – dans notre cas une structure anatomique – on construit une structure de données contenant les notions de sommets, d'arêtes et de tétraèdres. Pour chaque sommet, nous stockons la liste de ses tétraèdres voisins ainsi que celle de ses arêtes adjacentes, sa position au repos $\mathbf{P}_p^0$, sa position courante $\mathbf{P}_p$ et son tenseur de rigidité $\mathcal{B}^{pp}$ (équation 1.11). Chaque arête connaît ses deux sommets voisins ainsi que la liste des tétraèdres qui lui sont adjacents, et contient son tenseur de rigidité $\mathcal{B}^{pj}$. Enfin pour chaque tétraèdre, nous stockons ses quatre sommets, ses six arêtes, les constantes l'élasticité λ et μ et les quatre vecteurs de forme α_k. Il est aussi intéressant d'avoir une notion de triangle pour représenter la surface du maillage. Dans ce cas, chaque face externe connaîtra le tétraèdre auquel elle appartient et ses trois sommets. Cette notion de triangulation de surface permet en particulier de calculer de manière plus efficace les normales à la surface qui seront utiles pour afficher l'objet. Elles serviront aussi lors de la modélisation des contacts avec les outils chirurgicaux.

Le principe de fonctionnement du modèle masse-tenseur est relativement simple. À chaque instant t, on applique éventuellement de nouvelles conditions aux bords, puis on calcule les forces élastiques exercées sur chacun des sommets du maillage en utilisant l'équation 1.12. On intègre alors, pour chaque sommet, les équations du mouvement pour trouver sa nouvelle position à l'instant $t+1$.

3.2 Les conditions aux limites

Comme pour le modèle précédent, il faut prévoir un ensemble de conditions aux bords qui permettront de fixer ce modèle ou d'interagir avec lui :

- On peut fixer un sommet. Il suffit pour cela de ne pas tenir compte des forces externes et internes qu'il

subit et de ne jamais calculer sa nouvelle position.

- Pour pré-contraindre le modèle, on peut aussi déplacer un sommet avant de le fixer.
- On peut appliquer un déplacement sur un ou plusieurs sommets.
- On peut aussi appliquer des forces que l'on ajoute simplement aux forces internes avant d'intégrer les équations du mouvement. On distingue alors plusieurs types de forces : les forces instantanées qui ne durent que le temps d'une itération, les forces permanentes qui sont ré-appliquées à chaque itération, et les champs de forces qui correspondent à des forces permanentes s'appliquant sur l'ensemble du maillage.

3.3 Le calcul des forces

Afin de minimiser le nombre d'opérations, on n'applique pas directement la formule 1.12. On commence par parcourir les arêtes du maillage. Pour chaque arête $(\mathbf{P}_p,\mathbf{P}_j)$, on calcule la force qu'elle exerce sur ses deux sommets : $[\mathcal{B}^{pj}] * \mathbf{U}_j$ pour le sommet $\mathbf{P}_p$ et $[\mathcal{B}^{pj}]^t * \mathbf{U}_p$ pour le sommet $\mathbf{P}_j$. Ensuite, on parcourt l'ensemble des sommets. Pour chaque sommet $\mathbf{P}_p$, on calcule la force due à son déplacement : $[\mathcal{B}^{pp}] * \mathbf{U}_p$. On peut alors intégrer l'équation du mouvement pour trouver la nouvelle position du sommet.

3.4 L'intégration du mouvement

Nous considérons que l'équation qui gouverne le mouvement de notre modèle élastique linéaire est une équation différentielle Newtonienne de la forme :

$$m_i \frac{d^2\mathbf{P}_i}{dt^2} = \gamma_i \frac{d\mathbf{P}_i}{dt} + \mathbf{F}_i \tag{3.1}$$

Cette équation est reliée aux équations différentielles rencontrées en mécanique des milieux continus :

$$M\ddot{\mathbf{U}} + C\dot{\mathbf{U}} + K\mathbf{U} = \mathbf{R} \tag{3.2}$$

D'après la théorie des éléments finis, la matrice de masse M et la matrice d'amortissement C sont creuses et dépendent des propriétés de chaque tétraèdre. Dans notre cas, nous considérons que M et C sont des matrices diagonales, c'est à direque la masse et les effets d'amortissement sont concentrés sur les sommets du maillage. Cette simplification, appelée *masse-lumping*, découple les mouvements de chacun des nœuds et permet donc d'écrire l'équation 3.2 comme un ensemble d'équations différentielles indépendantes du type 3.1, une pour chaque sommet.

De plus, nous avons choisi un schéma d'intégration explicite, dans lequel les forces élastiques sont estimées à l'instant t pour calculer la nouvelle position du sommet à l'instant $t+1$:

$$\left(\frac{m_i}{\Delta t^2} - \frac{\gamma_i}{2\Delta t}\right)\mathbf{P}_i^{t+1} = \mathbf{F}_i + \frac{2m_i}{\Delta t^2}\mathbf{P}_i^t - \left(\frac{m_i}{\Delta t^2} + \frac{\gamma_i}{2\Delta t}\right)\mathbf{P}_i^{t-1} \tag{3.3}$$

Les schémas explicites sont seulement *conditionnellement stables*, ils convergent donc moins vite que les schémas implicites. Pour que le schéma soit stable, le pas de temps utilisé pour l'intégration numérique doit

être suffisamment petit. Le pas de temps critique dépend de la plus grande valeur propre de la matrice de rigidité et des valeurs locales de la masse et de l'amortissement. Pour optimiser le pas de temps, nous prenons la matrice de masse égale à l'identité, et nous cherchons à optimiser les valeurs locales de l'amortissement afin de limiter les risques d'oscillations. Ces opérations ont pour effet d'améliorer le conditionnement du système linéaire, et donc d'en stabiliser la résolution.

3.5 Changements de topologie du maillage

Une des tâches de base, que l'on veut être capable de faire avec un simulateur de chirurgie, consiste à découper les tissus mous. D'autre part, pour des raisons de précision de calcul, ou de finesse de la découpe, on peut être amené à modifier localement la topologie du maillage, par exemple en augmentant le nombre de tétraèdres dans une zone d'intérêt. Sa structure de données et le caractère local du calcul des forces, font du modèle masse-tenseur un outil particulièrement bien adapté pour simuler ce type d'opérations.

3.6 Performances et exemples

Bien évidement le modèle masse-tenseur est plus long à converger vers une solution que le modèle pré-calculé, pour la simple raison que tous les calculs sont effectués pendant la simulation. Par contre, on atteint assez vite (au bout de quelques itérations seulement), une solution rapprochée. Cette solution sera relativement exacte dans la zone où les contraintes sont appliquées. Par contre la déformation globale du modèle sera plus lente, car les contraintes doivent se propager de proche en proche dans tout le modèle. On évalue donc les performances de ce modèle par le nombre d'itérations que l'on peut faire entre deux affichages, le minimum étant d'en faire une, et donc d'avoir un modèle dont la fréquence de calcul est de 20Hz. Cette limite est atteinte sur un pentium II 450MHz pour un maillage comportant environ 10000 tétraèdres.

Les figures 3.1 et 3.2 présentent les résultats obtenus lors de la suppression de tétraèdres sur des modèles synthétiques précontraints. Pour le cylindre, nous avons fixé les sommets de la face inférieure et avons appliqué des déplacements aux sommets de l'autre face. Nous avons alors supprimé certains tétraèdres, provoquant ainsi un déséquilibre du modèle, qui évolue alors de manière naturelle vers sa nouvelle forme. Nous avons procédé de manière identique sur le deuxième modèle. Les sommets situés sur le bord droit sont fixés et certains sommets du bord gauche ont subi un déplacement. On observe, sur ces deux exemples le comportement réaliste du modèle masse-tenseur lors de changements importants de topologie.

3.7 Conclusion

La lourdeur du calcul fait que ce modè n'est pas envisageable pour de grosses structures avec beaucoup de nœuds, dans le cadre de simulations avec retour de forces. Il faut donc trouver un moyen de combiner les avantages du modéle pré-calculé et du masse-tenseur dans un même modèle hybride. Ainsi dans la partie suivante, je décrirai le travail que j'ai réalisé durant mon stage. Il correspond à l'élaboration d'un algorithme et à son application au simulateur de chirurgie.

FIG. 3.1 – *Déformation et découpe d'un modèle masse-tenseur représentant un cylindre.*

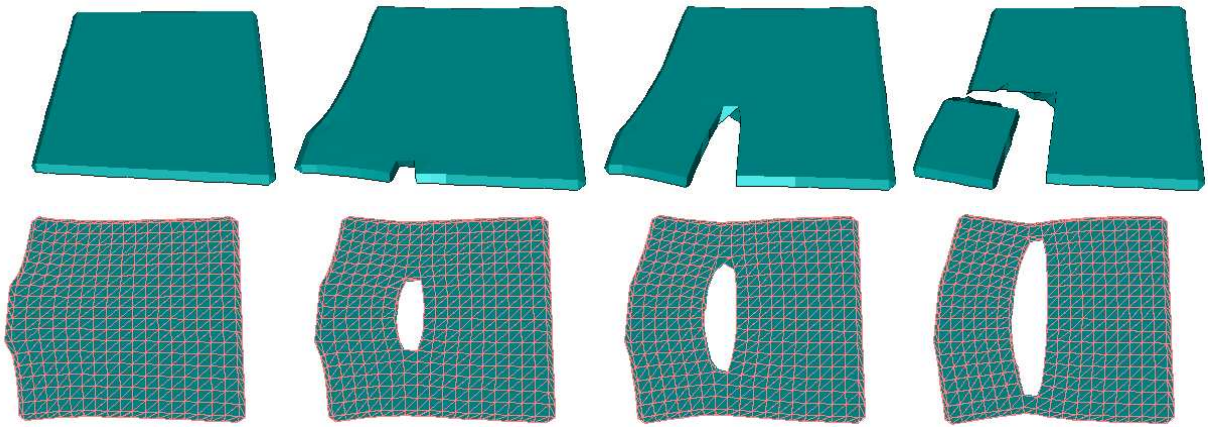


FIG. 3.2 – *Deux exemples de découpe sur un modèle précontraint. La déformation reste réaliste lors de la suppression des tétraèdres.*

Deuxième partie : Travail effectué pendant les cinq premiers mois de stage

Chapitre 4

Modèle hybride déformable

4.1 Introduction

Le but de ce chapitre est de présenter une méthode permettant d'intégrer, dans un même environnement de simulation, des modèles dynamiques déformables décrit au chapitre 2.5 et, ceux ayant un comportement statique évoqués au chapitre 1.3.

Deux objectifs principaux ont été fixés pour mon stage :

– le premier est de **faire cohabiter au sein d'un même maillage des zones avec des méthodes de résolutions différentes** comme représenté sur la figure 4.1;

– le deuxième, est de **faire passer dynamiquement un sous-domaine pré-calculé vers un sous-domaine masse-tenseur**. Ainsi, un organe sera composé au début de la simulation par un ensemble de sous-domaines pré-calculés, et, au fur et à mesure que le chirurgien voudra découper une zone, celle-ci se transformera en masse-tenseur afin d'autoriser la découpe du maillage.

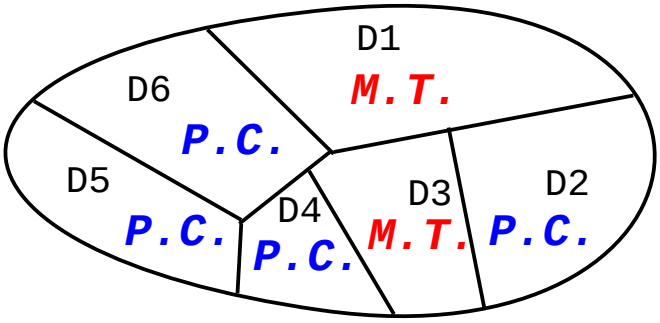


FIG. 4.1 – *Domaine découpé en plusieurs sous-domaines avec des lois d'évolution différentes (P.C.=Pré-calculé et M.T.=Masses-Tenseurs)*

Ce chapitre est inspiré en grande partie du travail effectué par S. Cotin et Al ([1] et [3]) fondé sur des travaux concernant la résolution d'équations aux dérivées partielles posées sur des domaines de géométrie

complexe : la technique de la **décomposition de Domaine** .

Le modèle hybride ainsi défini peut avoir des applications intéressantes dans le domaine de la simulation : apporter un comportement dynamique à un modèle statique élastique linéaire ou bien permettre de prendre en compte différents types d'interactions sur des sous domaines du modèle initial. Pour la simulation de chirurgie, cette technique va permettre des comportements de type découpe ou déchirement sur une partie du domaine tout en conservant un modèle pré-calculé pour le reste du maillage. Les temps de calculs devraient ainsi fortement accélérés et permettre des fréquences de rafraîchissement compatibles avec le retour de forces. Il devrait autoriser, par exemple, soit d'utiliser des lois de déformation plus sophistiquée, soit d'augmenter le nombre de nœuds dans notre modèle, ceci afin d'augmenter les détails et le réalisme du simulateur.

4.2 Théorie de la décomposition de domaine : les différentes solutions

Cette technique, introduite pour la résolution de problèmes complexes sur des domaines comportant un très grand nombre de nœuds repose, à l'origine, sur une résolution numérique parallélisée. Elle correspond, essentiellement à une découpe du domaine en sous-domaines plus petits; ceci dans un but :

❶ soit de simplifier les calculs. Par exemple pour les méthodes de pré-conditionnement, la décomposition de domaine implique la subdivision du processus de résolution d'un système linéaire faisant intervenir un très grand nombre d'́equations en un ensemble de problèmes plus petits dont les solutions peuvent être utilisées pour produire un pré-conditionneur pour le système d'équations résultant de la discrétisation des EDP sur le domaine entier. Dans ce contexte, la décomposition de domaine porte essentiellement sur la méthode de résolution du système algébrique issu de la discrétisation.

❷ soit de paralléliser. La décomposition de domaine est en effet idéalement adaptée au calcul parallèle ; l'existence et la création de différents sous-domaines représentent une façon naturelle de répartir le problème sur des architectures multiprocesseurs.

❸ soit (et c'est ce qui va nous intéresser particulièrement dans un premier temps) d'utiliser des modèles physiques différents sur chaque sous-domaines. Dans ce contexte, la décomposition de domaine fait plutôt référence au choix des équations aux dérivées partielles à résoudre sur chaque sous domaine.

Un point important à noter est que dans toutes les méthodes de décomposition de domaine, les deux étapes importantes à résoudre sont : la **détermination des sous-domaines** et la **communication entre ces sous-domaines** via des conditions aux frontières artificielles.

Dans ce cas les métt hodes de décomposition de domaine sont des techniques générales de résolution d'EDP qui utilisent certaines propriétés inhérentes aux EDP pour obtenir des résolutions rapides. Pour des problèmes linéaires, ces méthodes peuvent être vues comme des pré-conditionneurs pour des techniques de résolution itératives telles que la méthode du gradient conjugué. Parmi les nombreuses méthodes de décomposition de domaine, on peut les cataloguer ainsi:

1. Tout d'abord sur le fait qu'il y ait ou non un recouvrement des domaines. On peut distinguer la **méthode de Schwarz**, faisant appel à des recouvrements de sous domaines, ou les **méthodes de Lions**, **Jacobi**,et de **Gauss-Seidel**, où les différents sous-domaines ne se recouvrent pas (cf. [5] et [6]).

2. Ensuite sur l'élément que l'on partitionne. Ainsi on peut faire une partitionnement de type "basé sommet" ou de type "basé élément" (cf. figure 4.2). Dans une approche "basé sommet" la frontière ne passe pas par les nœuds mais par les arrêtes. Tandis que dans le cas d'une approche comme dans le simulateur en "basé élément" la frontière passe par les nœuds. Dans ce cas donc les nœuds situés à la frontière appartiennent à deux sous-domaines.

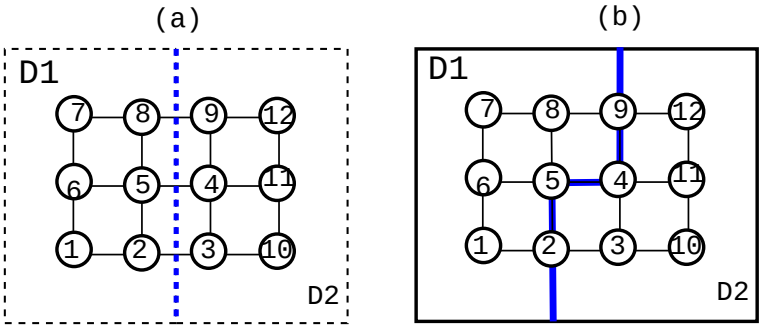


FIG. 4.2 – *(a) "basé sommet" et (b) "basé èlèment" partitionnement de 4x3 sommets en 2 sous-domaines.*

Le choix de ce partitionnement va influencer l'interprétation de la matrice **K** (cf. figure 4.3 et 4.4). On observe l'importance de la numérotation des nœuds des domaines au sein de la matrice **K**. Suivant la numérotation la forme de la matrice de rigidité n'est pas la même.

4.2.1 Décomposition de domaine avec recouvrement

En se basant sur [5] et [6], nous allons aborder la décomposition de domaine avec recouvrement. On va donc couper le problème initial en des sous-problèmes de petites tailles et sur des géométries plus simple mais qui se recouvrent. Pour illustrer nos propos, on considère trois sous-domaines placés comme dans la figure 4.5 où chaque domaine D_i étend sa limite dans les sous-domaines voisins. On appelle T_i la limite du domaine D_i et $T_{i,j}$ la limite du domaine i "étendue" dans le domaine j.

4.2.1.1 Algorithme de Schwarz (1890)

La première méthode de décomposition de domaine fut proposé par H. A. Schwarz dans une preuve d'existence pour résoudre un problème de Poisson ($\triangle.\mathbf{U} = 0$) sur un domaine de géométrie complexe. H. A. Schwarz propose une méthode alternée qui s'applique à la résolution de problèmes elliptiques sur un domaine composé de plusieurs sous-domaines tel que décrit à la figure 4.5.

Soit u_{ji} la restriction de la solution u sur la limite T_{ji}, la procédure alternée peut être décrite comme dans l'algorithme de la figure 4.7.

L'algorithme résout l'équation d'origine sur chaque sous-domaine avec certaines conditions aux limites, jusqu'à vérifier un critère d'arrêt. Par exemple lorsque $\| u_{ij}^{n+1} - u_{ij}^{n} \| < \varepsilon$ pour tout i et j. On remarque que les conditions aux limites dans ce cas sont assez simple, puisqu'il s'agit de conditions de Dirichlet.

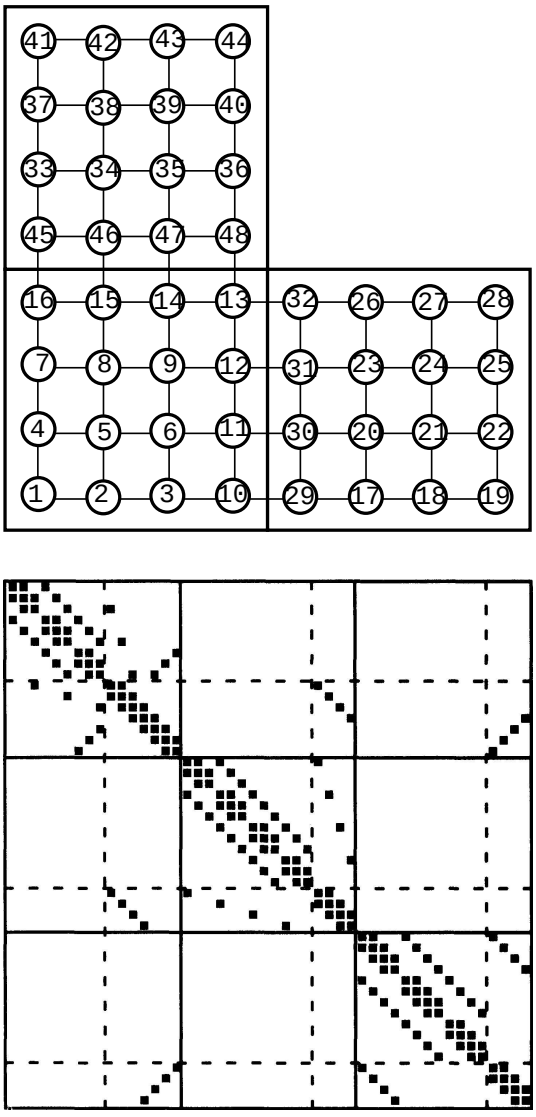


FIG. 4.3 – *Décomposition du type "basé sommet", du domaine en 3 sous-domaines et sa matrice* **K** *associée.*

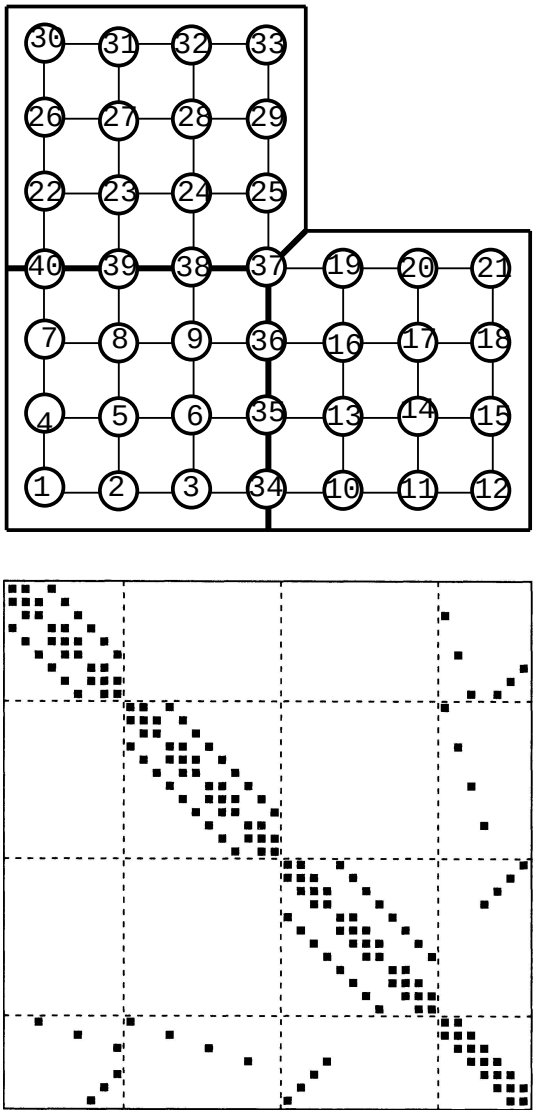


FIG. 4.4 – *Décomposition du type "basé èlèment", du domaine en 3 sous-domaines et sa matrice* **K** *associée.*

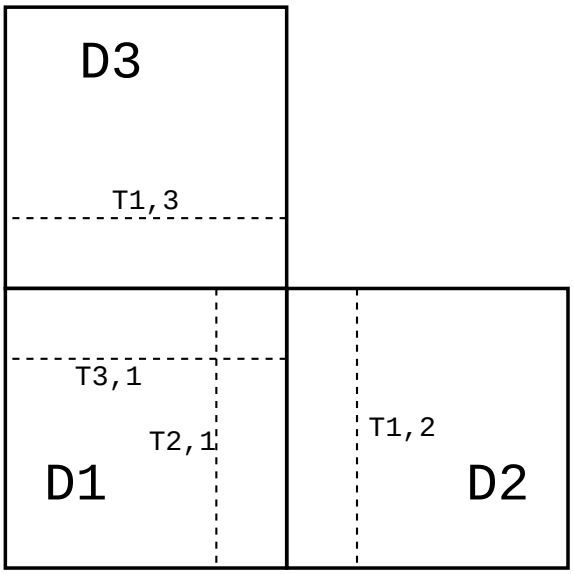


FIG. 4.5 – *Domaine en forme de L, divisé en 3 domaines D_1,D_2 et D_3 . (Schéma issue de [6])*

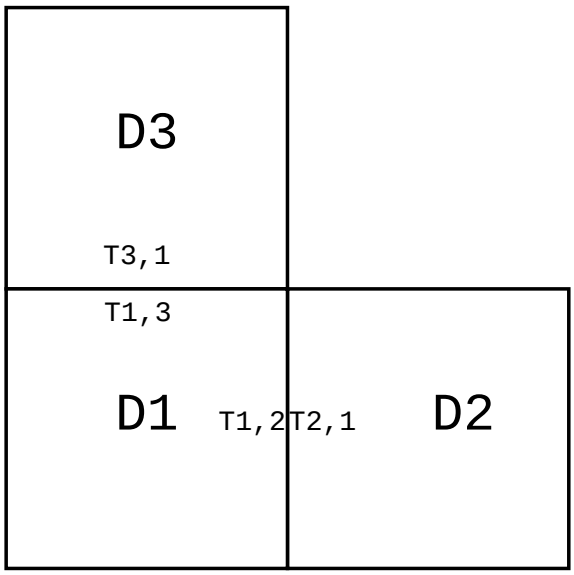


FIG. 4.6 – *Domaine en forme de L, divisé en 3 domaines D_1,D_2 et D_3 . (Schéma issue de [6])*

Algorithme :

---

1. Choix d'un u initial
2. jusqu' à la convergence :
3. POUR i = 1,....,s FAIRE :
4. Résoudre $\triangle u_i = f_i$ dans D_i avec $u = u_{ij}$ sur T_{ij}
5. Mettre a jour la valeur de u sur T_{ji}, pour tout j.
6. STOP_POUR
7. STOP_JUSQU'A

FIG. 4.7 – *Algorithme de Schwarz*

Algorithme :

---

1. Choix d'un u initial
2. jusqu'à la convergence :
3. POUR i = 1,....,s FAIRE :
4. Résoudre $\triangle u_i = f_i$ dans D_i avec $\frac{\partial u_i}{\partial \eta_{ij}} + \lambda_{ij}.u_i = -\frac{\partial u_j}{\partial \eta_j i} + \lambda_{ji}.u_j$ sur T_{ij}
5. STOP_POUR
6. STOP_JUSQU'A

FIG. 4.8 – *Algorithme de Lions*

4.2.1.2 Preuve de convergence

La convergence vers la solution globale de cet algorithme a été établie par Schwarz en utilisant le principe du Maximum. Ensuite vers les année 30, Sobolev a découvert la formulation variationnelle de l'algorithme de Schwarz pour des problèmes d'élasticité linéaire. Le résultat de Sobolev offre un point de départ pour utiliser l'algorithme pour des problèmes ne vérifiant pas le principe du maximum. Puis en 1988, P-L. Lions précise que la convergence de cette méthode dépend de la taille de recouvrement, du pas de discrétisation et du diamètre caractéristique des sous-domaines.

Bien entendu, plus le recouvrement est important, meilleure sera la convergence.

4.2.2 Décomposition de domaine sans recouvrement

Nous exposons ici, l'algorithme de décomposition de domaine sans recouvrement de P-L. Lions, qui est inspiré de la méthode originale de Schwarz ci-dessus. Dans ce cas, la décomposition de domaine correspond à la figure 4.6.

4.2.2.1 Algorithme de Lions

Si nous considérons le même problème que précédemment, l'algorithme de Lions est décrit dans la figure 4.8.

où $\eta_i j = -\eta_{ji}$ est le vecteur à la normal extérieur à D_i en la frontière non vide avec son voisin D_j et $\lambda_{ij} = \lambda_{ji}$ sont des réels strictement positif.

On observe, que cet algorithme est plus compliqué à mettre en oeuvre, dans la mesure où il fait intervenir des conditions de Neumann (dérivée partielle de u dans le sens de la normale à la limite) qui sont plus compliquées à calculer que les conditions de Dirichlet. C'est pourquoi d'autres techniques vont être utilisées.

4.2.2.2 Lien avec la décomposition de matrice par blocs

En faisant avec l'algorithme de Schwarz, l'hypothèse de linéarité de la discrétisation intérieure et des informations transférées d'un sous-domaine à l'autre, on obtient des algorithmes de type itération de Jacobi ou itération de Gauss-Seidel par bloc.

Si on partitionne nos matrices **K**, **F**, **U**, comme sur l'équation 4.1 :

$$\mathbf{K} = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \quad \mathbf{F} = \begin{bmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \end{bmatrix} \quad \mathbf{U} = \begin{bmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \end{bmatrix} \tag{4.1}$$

Le calcul itératif de **U** se fait de la façon décrite dans l'équation 4.2 :

$$\begin{cases} \mathbf{U}_1^{n+1} = \mathbf{K}_{11}^{-1}.(\mathbf{F}_1 - K_{12}.\mathbf{U}_2^n) \\ \mathbf{U}_2^{n+1} = \mathbf{K}_{22}^{-1}.(\mathbf{F}_2 - K_{21}.\mathbf{U}_1^{n+\nu}) \end{cases} \tag{4.2}$$

où $\nu = 0$ pour la méthode de Jacobi et $\nu = 1$ pour l'algorithme de Gauss-Seidel.

On notera que l'algorithme de Gauss-Seidel converge, en general, deux fois plus vite que les itérations de Jacobi (Consulter [5] pour plus de détails). Cependant ce premier ne sera pas parallélisable. Même si notre objectif premier n'est pas la parallélisation ces deux algorithmes seront implantés dans le simulateur.

4.2.2.3 Technique de condensation de matrice

Il est aussi intéressant de citer les techniques de condensation de matrice tel que le **complément de Schur** qui permet aussi une décomposition en bloc de la matrice. En effet, il est intéressant de voir ce que devient l'équation $\mathbf{K}.\mathbf{U} = \mathbf{F}$ si on décompose le domaine en deux sous domaines comme l'illustre la figure 4.10. En se basant sur la théorie précédente et sur les travaux de M. Bro-Nielsen et S. Cotin [4] sur la condensation, on montre que cette équation linéaire devient :

$$\begin{bmatrix} F_1 \\ F_2 \end{bmatrix} = \begin{bmatrix} K_1 & L_{12} \\ L_{21} & K_2 \end{bmatrix} . \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \tag{4.3}$$

où $\mathbf{F}_1$ et $\mathbf{U}_1$ (respectivement $\mathbf{F}_2$ et $\mathbf{U}_2$) sont les vecteurs des forces externes et des déplacements qui s'appliquent sur le domaine Ω_1 (resp. Ω_2) par l'intermédiaire $\mathbf{K}_1$ (resp. $\mathbf{K}_2$) qui est la matrice de raideur de son domaine.

$\mathbf{L}_{12}$ et $\mathbf{L}_{21}$ étant les matrices décrivant la relation des nœuds situés à l'interface des domaines.

Si on développe cette expression on obtient : $\boxed{\mathbf{F}_1 = \mathbf{K}_1.\mathbf{U}_1 + \mathbf{L}_{12}.\mathbf{U}_2}$ et $\boxed{\mathbf{F}_2 = \mathbf{K}_2.\mathbf{U}_2 + \mathbf{L}_{21}.\mathbf{U}_1}$ ce qui donne en rassemblant l'ensemble :

$$\begin{aligned} \mathbf{F}_1 &= \mathbf{K}_1.\mathbf{U}_1 + \mathbf{L}_{12}.\mathbf{K}_2^{-1}.\mathbf{F}_2 - \mathbf{L}_{12}.\mathbf{K}_2^{-1}.\mathbf{L}_{21}.\mathbf{U}_1 \\ \mathbf{F}_1 &= \mathbf{K}_1.\mathbf{U}_1 + \mathbf{L}_{12}.(\mathbf{K}_2^{-1}.(\mathbf{F}_2 - \mathbf{L}_{21}.\mathbf{U}_1)) \end{aligned} \tag{4.4}$$

$$\begin{aligned} \mathbf{F}_2 &= \mathbf{K}_2.\mathbf{U}_2 + \mathbf{L}_{21}.\mathbf{K}_1^{-1}.\mathbf{F}_1 - \mathbf{L}_{21}.\mathbf{K}_1^{-1}.\mathbf{L}_{12}.\mathbf{U}_2 \\ \mathbf{F}_2 &= \mathbf{K}_2.\mathbf{U}_2 + \mathbf{L}_{21}.(\mathbf{K}_1^{-1}.(\mathbf{F}_1 - \mathbf{L}_{12}.\mathbf{U}_2)) \end{aligned} \tag{4.5}$$

POUR LE DOMAINE NUMÉRO 1

1. Calcul de $\mathbf{K}_1^*$
2. Calcul de $\mathbf{U}_1^*$, le déplacement dû à $\mathbf{F}_1$ sur Ω_1 :
$$\mathbf{U}_1^* = \mathbf{K}_1^{-1}.\mathbf{F}_1 \qquad (4.8)$$
3. Calcul de $\mathbf{F}_1^* = \mathbf{F}_1 - \mathbf{L}_{12}.U_2^*$
4. Résolution de $\mathbf{F}_1^* = \mathbf{K}_1^*.\mathbf{U}_1$

POUR LE DOMAINE NUMÉRO 2

1. Calcul de $\mathbf{K}_2^*$ (Rq. : qui correspond au complément de Schur de la matrice dans l'équation 4.3).
2. Calcul de $\mathbf{U}_2^*$, le déplacement dû à $\mathbf{F}_2$ sur Ω_2 :
$$\mathbf{U}_2^* = \mathbf{K}_2^{-1}.\mathbf{F}_2 \qquad (4.9)$$
3. Calcul de $\mathbf{F}_2^* = \mathbf{F}_2 - \mathbf{L}_{21}.U_1^*$
4. Résolution de $\mathbf{F}_2^* = \mathbf{K}_2^*.\mathbf{U}_2$

FIG. 4.9 – *Algorithme de condensation de matrice*

Ces équations peuvent se mettre sous la forme équivalente à l'article [4] :

$$\boxed{\mathbf{F}_1^* = \mathbf{K}_1^*.\mathbf{U}_1} \quad ou \quad \begin{cases} \mathbf{F}_1^* = \mathbf{F}_1 - \mathbf{L}_{12}.\mathbf{K}_2^{-1}.\mathbf{F}_2 \\ \mathbf{K}_1^* = \mathbf{K}_1 - \mathbf{L}_{12}.\mathbf{K}_2^{-1}.\mathbf{L}_{21} \end{cases} \qquad (4.6)$$

$$\boxed{\mathbf{F}_2^* = \mathbf{K}_2^*.\mathbf{U}_2} \quad ou \quad \begin{cases} \mathbf{F}_2^* = \mathbf{F}_2 - \mathbf{L}_{21}.\mathbf{K}_1^{-1}.\mathbf{F}_1 \\ \mathbf{K}_2^* = \mathbf{K}_2 - \mathbf{L}_{21}.\mathbf{K}_1^{-1}.\mathbf{L}_{12} \end{cases} \qquad (4.7)$$

Cet algorithme (cf. figure 4.9) nécessite donc 2 itérations pour résoudre le système sur l'ensemble des domaines. Ceci implique donc un sur-coût de calcul qu'on ne peut pas concevoir avec des modèles du type masse-tenseur(M.T.) qui sont déjà source de problèmes pour le temps de calcul. Par contre en ce qui concerne le modèle pré-calculé (P.C.) et plus particulièrement une frontière P.C./P.C., ce système semble tout a fait intéressant par rapport à la méthode de Schwarz où la convergence sera beaucoup plus longue.

Ainsi dans le cas d'une frontière P.C./P.C. il faudra effectuer deux pré-calculs pour chaque domaine :

– le pré-calcul pour la matrice $\mathbf{K}_i^*$
– le pré-calcul pour le calcul de $\mathbf{U}_i^*$

4.2.3 Conclusion

Maintenant que nous avons fait un survol de la théorie, il reste a faire un choix parmi ces algorithmes et les modifier, puis les appliquer ou les appliquer directement au simulateur. Pour cela nous allons dans la prochaine section analyser le problème et ensuite, énumérer l'ensemble des solutions possibles.

4.3 Théorie de la décomposition de domaine: les solutions choisies

Pour respecter nos objectifs, il faut intégrer, dans un même modèle, plusieurs sous-domaines suivant une loi d'évolution différente (statique ou dynamique), et ceci de façon la plus générale possible, mais aussi le plus dynamique possible (chaque domaine doit pourvoir changer, sous certaines conditions, d'une loi d'évolution à une autre). Si on fait un grossissement du modèle représenté à la figure 4.1, l'équivalence, du point de vue

géométrique, entre la décomposition de domaine et un modèle hybride est illustrée par la figure 4.10. On y montre une approche "baée éléments".

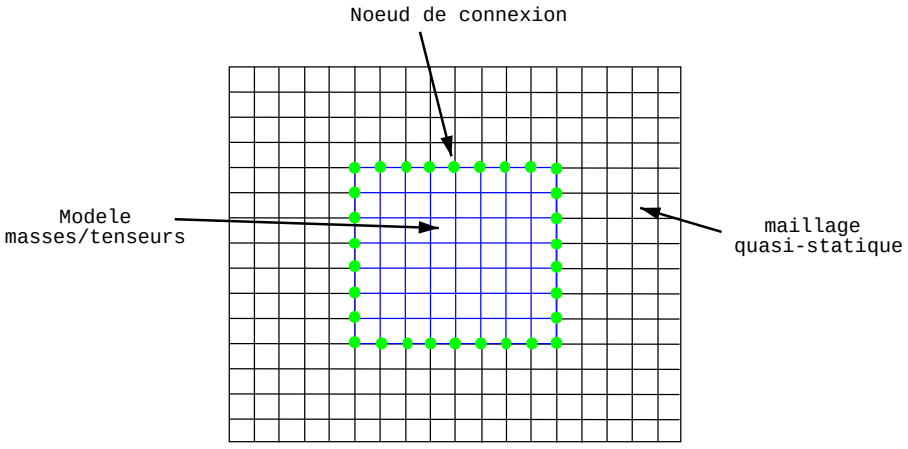


FIG. 4.10 – *Ce schéma correspond en quelque sorte à un grossissement de la figure 4.1. Le domaine correspondant au solide déformable est décomposé en deux sous domaines Ω_1 et Ω_2. La présence de nœuds au niveau de l'interface traduit une approche "basé èlèment".*

4.3.1 Analyse du problème

Maintenant que nous avons décrit notre modèle hybride et les techniques de décomposition de domaine, il faut essayer d'adapter la décomposition à nos deux lois de résolution (pré-calculé et masse-tenseur) et au simulateur déjà existant. En effet la théorie, vu au paragraphe précédent, résout une E.D.P. sur chaque sous-domaine en utilisant la même loi sur tout les domaines. La communication entre ces domaines est donc grandement facilitée. Dans notre cas, non seulement les méthodes nous sont imposées (pré-calculé et masse-tenseur), mais en plus ces méthodes n'utilisent pas forcément les mêmes informations et les mêmes conditions aux limites (cf. figure 4.11).

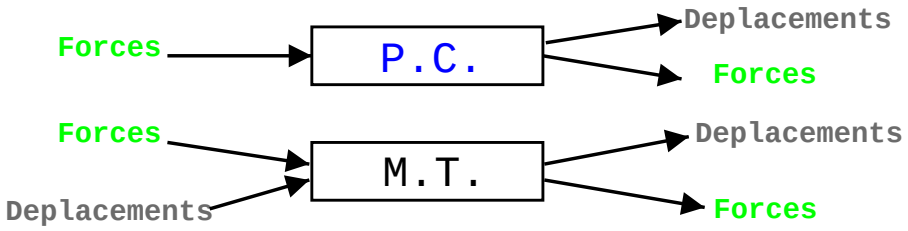


FIG. 4.11 – *Analyse systémique du problème (M.T.=Masse-Tenseur, P.C.=Pré-Calculé).*

Ainsi actuellement le modèle pré-calculé prend comme "argument" des forces et fournit des déplacements, (voire des forces dans certains cas). Alors que le modèle masse-tenseur transforme des forces ou des déplacements en des déplacements et des forces.

Nous allons voir maintenant quels algorithmes vus précédemment, peuvent nous aider.

4.3.2 Choix des algorithmes

Au sein de notre modèle hybride, nous allons rencontrer 3 types de frontières :

- **masse-tenseur / masse-tenseur**
- **pré-calculé / pré-calculé**
- **masse-tenseur / pré-calculé**

Pour chaque frontière il va falloir faire un choix (et justifier ce choix) parmi les nombreux algorithmes précédents.

Cas des algorithmes sans recouvrement Les méthodes de décomposition de domaine sans recouvrement, font appel à des conditions de Neumann qui sont très difficiles à mettre en oeuvre à cause du calcul de la dérivée partielle selon la normal à la surface de la bordure du domaine. C'est pour cette raison que nous pouvons déjà **éliminer l'algorithme de Lions**, de notre analyse. Il en va de même pour l'**algorithme de condensation de matrice**. Cette approche possède deux problèmes :

- le premier vient de la lourdeur de la structure de données qui lui est associée. En effet le stockage des deux pré-calculs par sous-domaine, semblent compliqué. De plus, cet algorithme se complique fortement lorsque le nombre de sous-domaine augmente.
- Le deuxième problème est commun avec les algorithmes de type itération de Jacobi. Il correspond au type de partitionnement du domaine. En effet ces deux méthodes sont fondées sur une approche "basée sommets" alors que la structure de données implantée dans le simulateur est du type "basés éléments". C'est pour cette raison que nous proposons la méthode suivante qui est très inspirée de l'algorithme de Gauss-Seidel.

4.3.2.1 Frontière masse-tenseur / masse-tenseur

Bien qu'il s'agisse d'une frontière homogène (c'est à dire ayant la même loi des deux cotés), nous n'avons pas choisi d'utiliser les algorithmes précédents, afin de respecter la durée du stage. Nous avons donc décidé d'éliminer ce type de frontière en regroupant les sous-domaines masse-tenseur qui sont voisins. Cependant il serait, dans un future proche, intéressant de pouvoir gérer cette frontière afin d'autoriser une parallélisation du simulateur.

4.3.2.2 Frontière pré-calculé / pré-calculé

Dans le cas de frontières homogènes, on peut tout à fait utiliser des algorithmes de type itération de Jacobi ou de Gauss-Seidel. Cependant comme je l'ai souligné dans le paragraphe 4.3.2, ces méthodes ne correspondent pas à la structure de données implantée dans le simulateur qui est du type "basée élèments" (un noeud frontière appartient à deux sous-domaines). Donc, si on considère les nœuds comme élément, il y a forcément un recouvrement entre les zones d'au moins un nœud. Cependant, ces algorithmes s'appliquent particulièrement bien à des sous-domaines pré-calculés qui permettent le pré-calcul des déplacements mais aussi celui des forces à envoyer à l'interface de communication. C'est pourquoi nous allons utiliser l'algorithme de Gauss-Seidel légèrement modifié.

Algorithme de Gauss-Seidel avec recouvrement Cet algorithme est en fait l'adaptation de la méthode de Jacobi ou de Gauss-Seidel à la numérotation de la matrice **K** comme définie sur la figure 4.4.

Si on décompose les matrices **K**, **U** et **F** de la manière suivante :

$$\mathbf{K} = \begin{pmatrix} \mathbf{K}_{1_{int}} & 0 & 0 & \mathbf{L}_{1_{I\_Fr}} \\ 0 & \mathbf{K}_{2_{int}} & 0 & \mathbf{L}_{2_{I\_Fr}} \\ 0 & 0 & \mathbf{K}_{3_{int}} & \mathbf{L}_{3_{I\_Fr}} \\ \mathbf{L}_{1_{Fr\_I}} & \mathbf{L}_{2_{Fr\_I}} & \mathbf{L}_{3_{Fr\_I}} & \mathbf{L}_{Fr} \end{pmatrix} \quad \mathbf{U} = \begin{pmatrix} \mathbf{U}_{1_{int}} \\ \mathbf{U}_{2_{int}} \\ \mathbf{U}_{3_{int}} \\ \mathbf{U}_{Fr} \end{pmatrix} \quad \mathbf{F} = \begin{pmatrix} \mathbf{F}_{1_{int}} \\ \mathbf{F}_{2_{int}} \\ \mathbf{F}_{3_{int}} \\ \mathbf{F}_{Fr} \end{pmatrix} \tag{4.10}$$

où les matrices $\mathbf{K}_{i_{int}}$ font référence à tous les nœuds du domaine i excepté les nœuds situés à la frontière avec d'autres domaines. Et les matrices $L_{i_{I\_Fr}}$ et $L_{i_{Fr\_I}}$ sont les relations entre ces nœuds frontières et les nœuds à l'intérieur du domaine.

L'élément $\mathbf{L}_{Fr}$ correspond à l'ensemble des nœuds et arêtes frontières de tous les sous domaines du maillage. On peut donc le décomposer comme la somme de chaque nœuds et arrêtes frontières de chaque sous-domaine (cf .équation 4.11).

$$\mathbf{L}_{Fr} = \sum \mathbf{L}_{Fr_i} \tag{4.11}$$

Il convient de faire un peu attention à ce terme car il est global. Dans certains par exemple quand un nœuds appartient à plus de deux sous-domaines, il peut correspondre à une force á transmettre aux sous-domaines et donc se retrouver dans le terme de droite dans l'équation 4.14.

L'équation $\mathbf{F} = \mathbf{K}.\mathbf{U}$ devient :

$$\begin{cases} \mathbf{K}_{1_{int}}.\mathbf{U}_{1_{int}} + \mathbf{L}_{1_{I\_Fr}}.\mathbf{U}_{Fr} = \mathbf{F}_{1_{int}} \\ \mathbf{K}_{2_{int}}.\mathbf{U}_{2_{int}} + \mathbf{L}_{2_{I\_Fr}}.\mathbf{U}_{Fr} = \mathbf{F}_{2_{int}} \\ \mathbf{K}_{3_{int}}.\mathbf{U}_{3_{int}} + \mathbf{L}_{3_{I\_Fr}}.\mathbf{U}_{Fr} = \mathbf{F}_{3_{int}} \\ \mathbf{L}_{1_{Fr\_I}}.\mathbf{U}_{1_{int}} + \mathbf{L}_{2_{Fr\_I}}.\mathbf{U}_{2_{int}} + \mathbf{L}_{3_{Fr\_I}}.\mathbf{U}_{3_{int}} + \mathbf{L}_{Fr}.\mathbf{U}_{Fr} = \mathbf{F}_{Fr} \end{cases} \tag{4.12}$$

Ce qui donne pour le sous-domaine 1 :

$$\begin{cases} \mathbf{K}_{1_{int}}.\mathbf{U}_{1_{int}} + \mathbf{L}_{1_{I\_Fr}}.\mathbf{U}_{Fr_1} = \mathbf{F}_{1_{int}} \\ \mathbf{L}_{1_{Fr\_I}}.\mathbf{U}_{1_{int}} + \mathbf{L}_{Fr1}.\mathbf{U}_{Fr_1} = \mathbf{F}_{Fr_1} - \mathbf{L}_{2_{Fr\_I}}.\mathbf{U}_{2_{int}} - \mathbf{L}_{3_{Fr\_I}}.\mathbf{U}_{3_{int}} - \sum_{i\neq 1} \mathbf{L}_{Fr_i}.\mathbf{U}_{Fr_i} \end{cases} \tag{4.13}$$

En mettant ce système d'équations sous forme matricielle et en supposant, pour simplifier, qu'il n'y a pas de cas particulier ($\sum_{i\neq 1} \mathbf{L}_{Fr_i}.\mathbf{U}_{Fr_i} = 0$):

$$\begin{pmatrix} \mathbf{K}_{1_{int}} & \mathbf{L}_{1_{I\_Fr}} \\ \mathbf{L}_{1_{Fr\_I}} & \mathbf{L}_{Fr_1} \end{pmatrix} . \begin{pmatrix} \mathbf{U}_{1_{int}} \\ \mathbf{U}_{Fr_1} \end{pmatrix} = \begin{pmatrix} \mathbf{F}_{1_{int}} \\ \mathbf{F}_{Fr_1} \end{pmatrix} - \begin{pmatrix} \mathbf{0} \\ \mathbf{L}_{2_{Fr\_I}}.\mathbf{U}_{2_{int}} \end{pmatrix} - \begin{pmatrix} \mathbf{0} \\ \mathbf{L}_{3_{Fr\_I}}.\mathbf{U}_{3_{int}} \end{pmatrix}. \tag{4.14}$$

En regroupant les 3 premières matrices, on obtient l'équation suivante :

$$\mathbf{K}_1.\mathbf{U}_1 = \mathbf{F}_1 - \begin{pmatrix} \mathbf{0} \\ \mathbf{L}_{2_{Fr\_I}}.\mathbf{U}_{2_{int}} \end{pmatrix} - \begin{pmatrix} \mathbf{0} \\ \mathbf{L}_{3_{Fr\_I}}.\mathbf{U}_{3_{int}} \end{pmatrix} \tag{4.15}$$

On peut donc maintenant appliquer la forme itérative de l'algorithme de Gauss-Seidel :

$$\begin{aligned}
\mathbf{K}_1.\mathbf{U}_1^{n+1} &= \mathbf{F}_1^{n+1} - \begin{pmatrix} \mathbf{0} \\ \mathbf{L}_{2_{Fr_I}}.\mathbf{U}_{2_{int}}^{n} \end{pmatrix} - \begin{pmatrix} \mathbf{0} \\ \mathbf{L}_{3_{Fr_I}}.\mathbf{U}_{3_{int}}^{n} \end{pmatrix} \\
\mathbf{K}_2.\mathbf{U}_2^{n+1} &= \mathbf{F}_2^{n+1} - \begin{pmatrix} \mathbf{0} \\ \mathbf{L}_{1_{Fr_I}}.\mathbf{U}_{1_{int}}^{n+1} \end{pmatrix} - \begin{pmatrix} \mathbf{0} \\ \mathbf{L}_{3_{Fr_I}}.\mathbf{U}_{3_{int}}^{n} \end{pmatrix} \\
\mathbf{K}_3.\mathbf{U}_3^{n+1} &= \mathbf{F}_3^{n+1} - \begin{pmatrix} \mathbf{0} \\ \mathbf{L}_{2_{Fr_I}}.\mathbf{U}_{2_{int}}^{n+1} \end{pmatrix} - \begin{pmatrix} \mathbf{0} \\ \mathbf{L}_{1_{Fr_I}}.\mathbf{U}_{1_{int}}^{n+1} \end{pmatrix}
\end{aligned} \tag{4.16}$$

Cet algorithme converge évidemment moins vite qu'un algorithme de condensation. Par contre, il offre l'avantage d'avoir une structure de données beaucoup plus simple et d'utiliser une partie du code existant du simulateur. En effet, on remarque que les termes que l'on additionne au membre de gauche correspondent uniquement aux forces qu'exercent les nœuds intérieurs d'un sous-domaine, sur les nœuds situés à la frontière du sous-domaine (cf. figure 4.12). Ce transfert d'informations d'un sous-domaine à un autre est donc facile à implémenter.

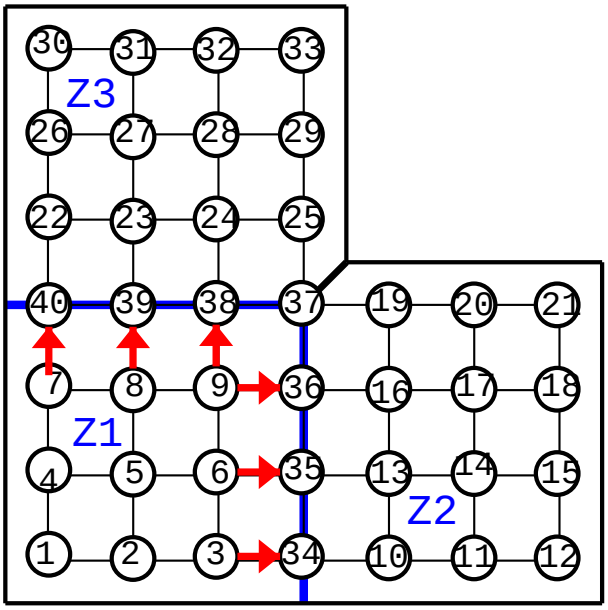


FIG. 4.12 – *Transfert d'informations d'un sous-domaine à un autre sous forme de forces, qu'exercent les nœuds intérieurs d'un sous-domaine, sur les nœuds situés à la frontière du sous-domaine*

4.3.2.3 Frontière masse-tenseur / pré-calculé

Dans le cas de cette frontière hétérogène, les choses se compliquent un peu, dans la mesure où les algorithmes précédents ne s'appliquent pas directement. Nous avons choisi d'utiliser un algorithme de type de Schwarz pour sa simplicité et pour son recouvrement. Le seul point qu'il reste encore à définir est la taille de recouvrement des domaines.

Comme indiqué dans le chapitre sur la théorie, plus le recouvrement est important, meilleure est la convergence. Il est donc intéressant de connaître l'influence de la taille du recouvrement. Néanmoins, on peut distinguer deux cas :

- **le cas normal où le recouvrement est assez grand** (supérieur ou égal à deux nœuds de large).
- **le cas où le recouvrement vaut un nœud de large** (cf. figure 4.10). Cette possibilité est très intéressante, car elle permet de ne gérer l'information entre les sous-domaines, qu'au niveau d'un nœud et ceci pour les deux cotés en même temps. Ce cas extrême est possible par une astuce de calcul que S. Cotin a développé dans sa thèse (cf. [1]). D'un calcul sur un domaine à l'autre on va soit :
  - fixer les forces sur les nœuds aux limites du domaine (cf. figure 4.10), et calculer les déplacements;
  - fixer les déplacements sur les nœuds aux limites du domaine (cf. figure 4.10), et calculer les forces.

  Ce système s'applique bien à une frontière de type M.T./P.C., comme le montre la figure 4.13 tirée de [1].

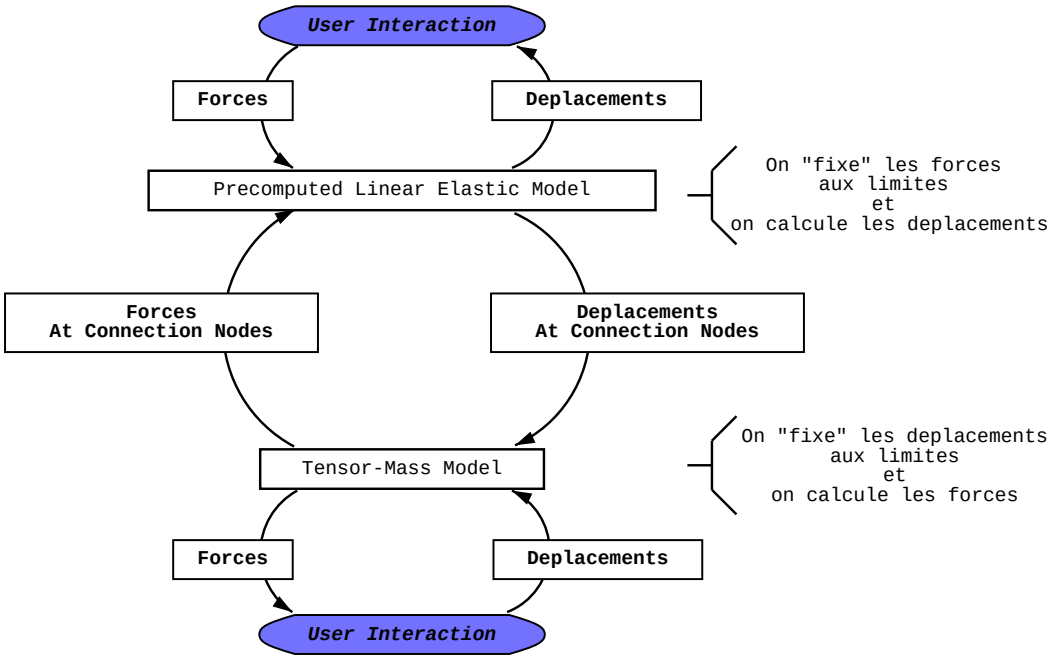


FIG. 4.13 – *boucle d'interaction du modèle hybride sur une frontière masse-tenseur/pré-calculé*

4.4 Mise en œuvre dans le simulateur des sous-domaines pré-calculés

L'architecture logiciel du simulateur étant assez élevée, il a été relativement difficile d'intégrer le modèle hybride. Il a fallu pour cela modifier légèrement l'algorithme déjà existant dans le simulateur pour que le pré-calcul porte, non seulement sur la nouvelle position du sous-domaine mais aussi sur les forces qu'il va envoyer aux zones voisines. Et tout ceci en respectant la structure de données déjà existante pour que l'intégration future du code au simulateur soit plus facile.

Nous avons testé successivement deux structures de données qui ont des complexités légèrement différentes :

4.4.1 Pré-calcul des nœuds à la surface et aux frontières

Dans cette première mise en œuvre, on ne pré-calcule la nouvelle position que pour les nœuds à la surface et à la frontière des sous-domaines. Par contre, ceci nous oblige à pré-calculer aussi les forces que l'on transmettra aux zones voisines.

Pour chaque sous-domaine ou zone il va y avoir :

- transformation des déplacements imposés, en forces, comme indiqué dans le chapitre 1.3,
- récupération des forces appliquées par l'utilisateur (instruments chirurgicaux),
- récupération des forces issus des sous-domaines voisins,
- calcul de nouvelles positions des noeuds (frontières + surface) résultantes des forces précédentes,
- calcul des forces aux frontières (pour envoyer aux domaines voisins) dû aux forces appliquées sur le domaine.

4.4.1.1 Taille du stockage de l'information

Dans cet algorithme on effectue 2 pré-calculs, l'un pour le déplacement des nœuds et l'autre pour le calcul des forces que l'on échange d'une zone à l'autre. Si on pose que pour une zone :

- **N** est le nombre total de nœuds du maillage,
- **S** est le nombre total de nœuds à la surface,
- **F** est le nombre total de nœuds à la frontière avec une autre zone,

alors le nombre de tenseurs 3x3 de pré-calculs stockés pour le déplacement est de $(\mathbf{F}+\mathbf{S})^2$ et pour les forces est de $\mathbf{F}.(\mathbf{F}+\mathbf{S})$. On stocke donc $(2.\mathbf{F}+\mathbf{S}).(\mathbf{F}+\mathbf{S})$ tenseurs en tout.

Remarque : On considère $(\mathbf{F}+\mathbf{S})$ et non **N** pour le calcul car les forces ne s'appliquent que sur les nœuds de surface et extérieurs.

4.4.1.2 Complexité des itérations

La complexité de cet algorithme est très difficile à évaluer car il dépend de la forme des sous-domaines et plus particulièrement du nombre de nœuds aux frontières et à la surface. Pour simplifier le calcul on va considérer :

- qu'il n'y a pas de forces extérieures qui s'appliquent sur le maillage. Ce cas se produit par exemple lorsque le maillage revient à sa position de repos après avoir subi une déformation,
- et que l'opération de base que l'on utilise pour calculer la complexité, est la multiplication d'un tenseur 3x3 par un vecteur de taille 3.

Dans le cas de la première condition, la zone doit juste récupérer les forces des autres zones et mettre à jour la position des ces nœuds situés à la frontière et à la surface. Pour une zone, la complexité de son calcul vaut $\mathbf{F}.(\mathbf{F}+\mathbf{S})$ pour le calcul des déplacements dûs aux forces des zones voisines et vaut $\mathbf{F}^2$ pour le calcul des forces que l'on transmet aux autres zones. On fait donc pour cet algorithme $\mathbf{F}.(\mathbf{2}.F+\mathbf{S})$ opérations de base.

4.4.2 Pré-calcul de tous les nœuds

Dans cet algorithme on fait en quelque sorte l'inverse de ce qui est fait précédemment. On va calculer la position de tous les nœuds du maillage. Ceci va nous permettre de ne plus faire de pré-calculs pour les forces extérieures. En effet, ces forces correspondent à la force qu'exercent les nœuds à l'intérieur de la zone sur les nœuds frontières de cette même zone. Il est donc facile, puisque l'on connaît toutes les positions, de déduire cette force en fonction de la position des noeuds intérieurs.

Pour chaque sous-domaine ou zone il va y avoir :

– transformation des déplacements imposés, en forces, comme indiqué dans le chapitre 1.3,

– récupération des forces appliquées par l'utilisateur (instruments chirurgicaux),

– récupération des positions des nœuds situés dans les sous-domaines voisins,

– calcul des forces correpondantes à ces positions,

– calcul des nouvelles positions (de tous les noeuds de la zone) résultantes des forces précédentes,

4.4.2.1 Taille du stockage de l'information

En prenant les mêmes conventions que précédemment, on va stocker pour chaque zone uniquement le pré-calcul de déplacement mais ici pour l'ensemble des nœuds du sous-domaine. Donc, pour chaque zone, il faut stocker $\mathbf{N}.(\mathbf{F}+\mathbf{S})$ tenseurs 3x3.

4.4.2.2 Complexité

Si on prend les mêmes conventions que pour le calcul de complexité précédent, on a $\mathbf{F}.N$ opérations de base pour le calcul des positions. Si on suppose qu'un nœud a à peu prés 20 voisins, il y a $(20/2).\mathbf{F}$ opérations de base pour le calcul des forces à envoyer aux sous-domaines voisins. On fait donc, en tout, pour cet algorithme $\mathbf{F}.\mathbf{N}+10.\mathbf{F}$ opérations de base.

4.4.3 Conclusion

Pour qu'il y ait une réelle différence entre les deux algorithme il faut que sur une zone $\mathbf{N} >> 2.\mathbf{F}+\mathbf{S}$. Ceci est loin d'être le cas pour les maillages que nous utilisons actuellement dans le simulateur (foie avec 500 sommets). De plus cette difference diminue en fonction que le nombre de zone croit. Ainsi, pour un nombre de zone supérieur à 4, le modèle qui pé-calcule tout devient d'une compléxité plus faible.

C'est pourquoi nous avons choisi, de garder ce modéle qui possede en plus une plus grande simplicité dans l'implémentation du code.

Evidemment tous ces résultats sont à comparer avec le masse-tenseur qui est le plus lent des 2 modèles d'origine. En supposant que chaque nœud du maillage a à peu prés 20 nœuds voisins, la complexité dans ce cas est $20.N$ opérations élémentaires. Comparer ce résultat avec le premier algorithme est assez compliqué par contre la comparaison est très simple pour le deuxième. Dans le meilleur des cas (où il n'y pas de forces externes qui s'exercent sur le maillage), la complexité du cas 2 est de $\mathbf{F}.\mathbf{N}$ opérations élémentaires (on néglige le calcul de forces).On voit donc que si F est plus grand que 20 (ce qui est pratiquement toujours le cas) le masse-tenseur sera le plus rapide. De plus, les variables varient différemment en fonction de la taille du

maillage. En effet : N correspond à un volume et varie donc au cube, tandis que B et S correspondent à une surface et croissent au carré.

On remarque donc que la différence entre ces algorithmes est très mince et que finalement le modèle hybride apporte un léger gain qui est tout de suite compensé par la complexité et la lourdeur de l'échange de l'information d'une zone à une autre.

4.5 Implémentation dans le simulateur des sous-domaines masse-tenseur

Ces sous-domaines ont été faciles à mettre en œuvre dans le simulateur car aucune modification concernant le calcul de position a été nécessaire. Et grâce à l'implementation qui pré-calcule tout les nœuds, les sous-domaines pré-calculés voisins sont completement transparents pour la partie masse-tenseur du maillage. Seul le passage dynamique d'un sous-domaine pré-calculé à un sous-domaine masse-tenseur a été un peu plus compliqué dans la mesure où il a fallut gérer les modifications de la structure de données associées à ce changement (cf. figure 4.14).

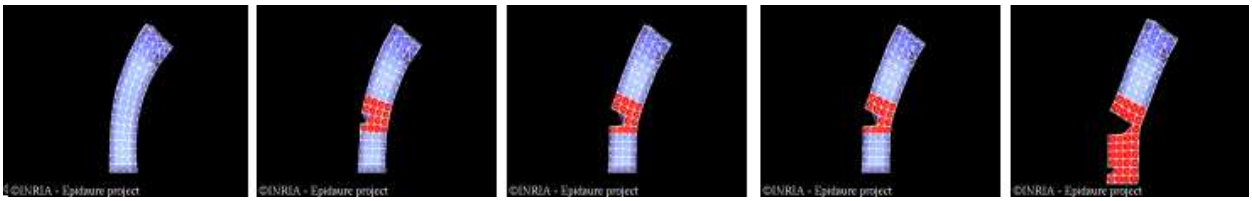


FIG. 4.14 – *changement de sous-domaines pré-calculés (bleu) en sous-domaines masse-tenseurs (rouge) au fur et à mesure de la découpe du maillage*

Chapitre 5

Résultats

Malheureusement, au moment où j'écris ce rapport, seuls les interfaces P.C./P.C. ont été testé dans le simulateur. Je ne peux donc que présenter les résultats concernant ce type d'interface. Lors de nos tests, nous avons essayé de tester, pour chaque cas, la convergence de l'algorithme mais aussi sa fréquence de fonctionnement.

Un bon modèle physique, doit pourvoir converger, afin d'obtenir la déformation correspondant aux lois élastiques que l'on utilise, mais il doit aussi avoir une fréquence de rafraîchissement suffisamment rapide afin de renseigner le système de retour de forces et l'affichage dans des temps raisonnables. Tous les résultats de cette section ont été obtenus avec un ordinateur de type intel de fréquence 1GHz.

5.1 Convergence

Après expérimentation, on observe que la convergence va dépendre de plusieurs facteurs :

- le nombre de zones,
- la taille de la zone sur laquelle on applique une contrainte,
- de l'application des contraintes sous la forme de forces ou de déplacements (comme nous le faisons dans le simulateur).

Les tests qui suivent ont été effectués sur une barre, représentée à la figure 5.1. Cette barre subie une déformation dû à une force (représenté en rouge) qui s'exerce sur un noeud.

5.1.1 Les contraintes sont des forces

Nous allons nous intéresser, dans un premier temps à la convergence dans le cas général, c'est à dire dans le cas où l'on applique uniquement des forces.

Convergence en fonction du nombre de zones On observe que notre algorithme converge. Ainsi, si on calcule l'erreur moyenne par rapport à une résolution rigoureuse de $\mathbf{F} = \mathbf{K}.\mathbf{U}$ qui donne la position à l'équilibre du maillage sous l'action d'une force (figure 5.1), on trouve une erreur assez petite au bout de 1000 itérations, que ce soit pour 2 sous-domaines ou plus (figure 5.2). Évidemment, plus il y a de sous-domaines, moins le

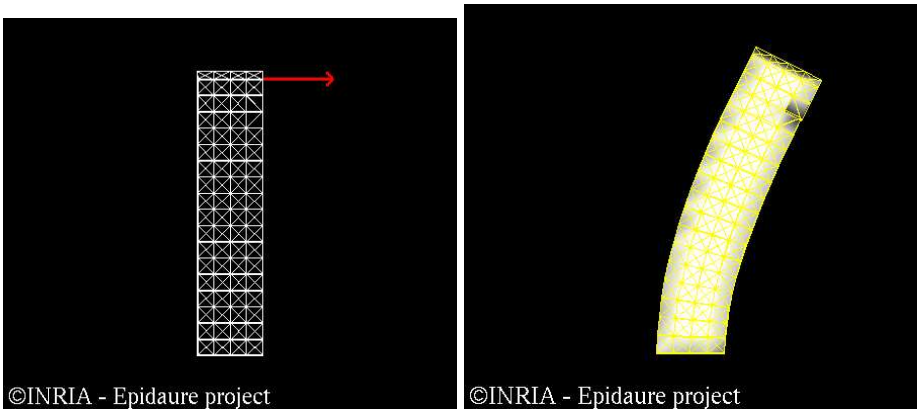


FIG. 5.1 – *Maillages utilisés pour faire les tests, avec l'action d'une force sur celui-ci et sa position d'équilibre. Ce maillage contient : 192 noeuds, 1612 arêtes, 1640 triangles et 820 tétraèdres.*

maillage converge vite. Ceci est dû aux transferts des forces d'un sous-domaine à un autre qui est plus long s'il y a plus de sous-domaines. Cependant la différence est minime.

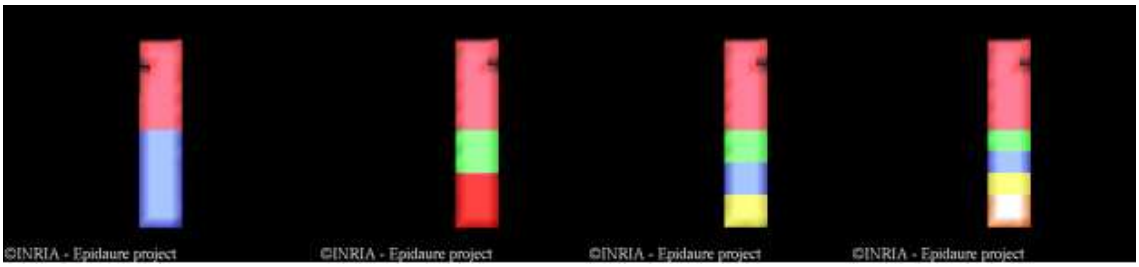


FIG. 5.2 – *Maillages utilisés pour faire le test avec 2, 3, 4 et 5 sous domaines.*

Cette erreur est représentée en fonction du nombre d'itérations, sur la figure 5.3 pour un maillage moyen avec plusieurs sous-domaines qui subissent l'action d'une force sur leur maillage. Cette erreur est normalisée par rapport à la longueur d'une arrête de notre maillage carré.

Convergence en fonction de la taille des sous-domaines Pour démontrer la relation entre la taille des zones et la convergence, nous avons utilisé trois types de maillages ayant chacun 2 zones et étant représentés sur la figure 5.4.

Comme le montre la figure 5.5, la vitesse de convergence dépend de la topologie du maillage. Plus la zone que subit la force est petite, plus la convergence sera lente. En effet si la zone qui subit une contrainte est très grande et correspond à quasiment à la totalité du maillage, la position d'équilibre va être atteinte presque immédiatement dés les premières itérations.

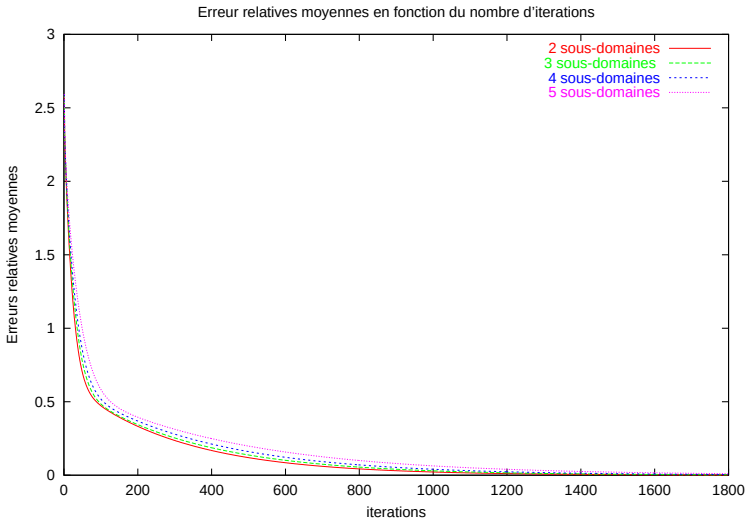


FIG. 5.3 – *Erreurs relatives (erreurs normalisées) lors de l'application d'une force à un maillage ayant plusieurs sous-domaines, en fonction du nombre d'itérations.*

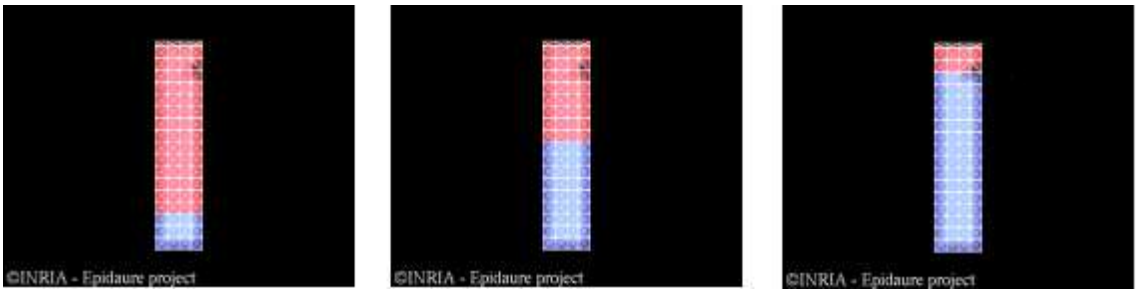


FIG. 5.4 – *Topologie des 3 maillages utilisés pour le test*

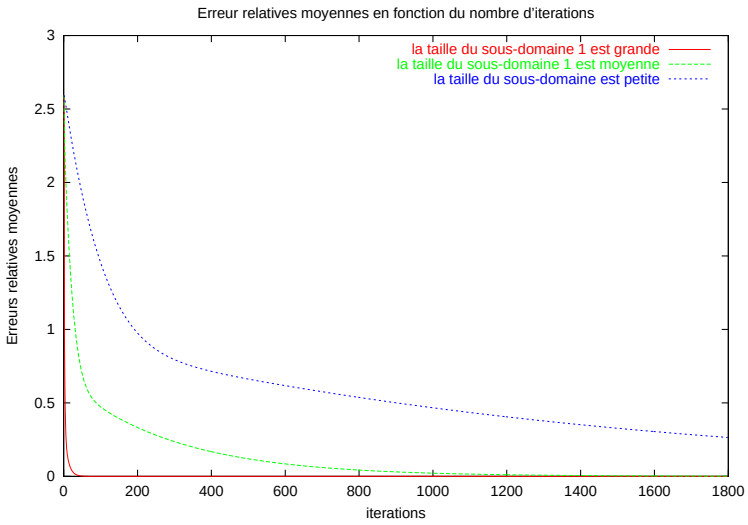


FIG. 5.5 – *Erreurs relatives (erreurs normalisées) lors de l'application d'une force à un maillage ayant 2 sous-domaines mais ayant des tailles de sous-domaines différentes, en fonction du nombre d'itérations.*

5.1.2 Les contraintes sont des déplacements

Dans cette section nous allons effectuer les mêmes tests que précédemment mais ici au lieu d'imposer une force nous allons faire, comme dans le simulateur, c'est à dire, imposer des déplacements à un nœud et nous allons regarder la convergence. Pour évaluer cette convergence nous allons calculer l'erreur (comme précédemment) par rapport à la résolution de $F = K.U$ qui correspond à l'équilibre représenté sur la figure 5.6.

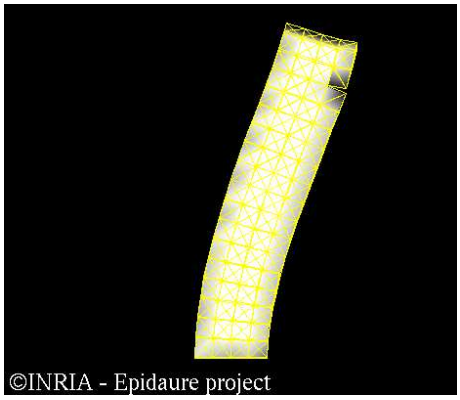


FIG. 5.6 – *Position du maillage obtenue par la résolution de $F = K.U$* avec un déplacement imposé sur un nœud.

Convergence en fonction du nombre de zones On trouve une erreur assez petite au bout de 200 itérations, que ce soit pour 2 sous-domaines ou plus (figure 5.7). Ceci est donc bien meilleur. Évidemment plus il y a de

sous-domaines, moins le maillage converge vite. Ceci est dû aux transferts des forces d'un sous-domaine à un autre qui est plus long s'il y a plus de sous-domaines. Cependant la différence est minime.

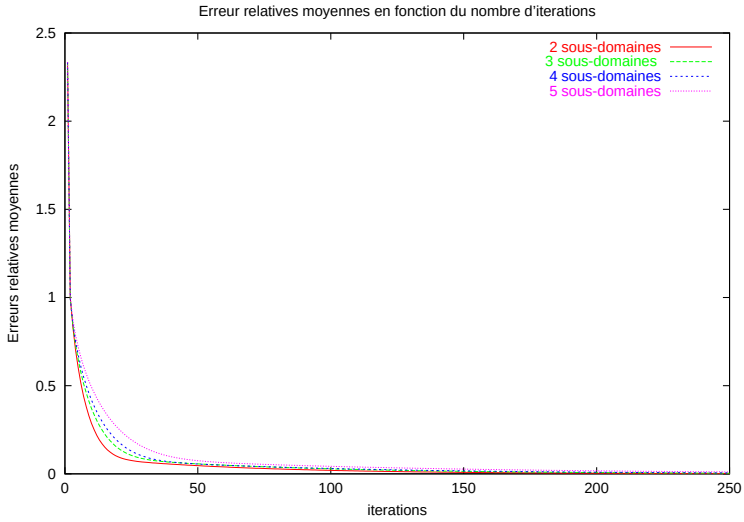


FIG. 5.7 – *Erreurs relatives (erreurs normalisées) lors de l'application d'un déplacement à un maillage ayant plusieurs sous-domaines , en fonction du nombre d'itération.*

Convergence en fonction de la taille des sous-domaines Comme le montre la figure 5.8, la vitesse de convergence dépend toujours de la topologie du maillage. Cependant la différence d'une topologie à l'autre est fortement diminué et la convergence est atteinte dans tous les cas au bout 100 itérations. Ceci montre que si on se place dans le cas de contraintes sous forme de déplacements la convergence est rapide.

Comme le montre le graphique sur la figure 5.9, la convergence est nettement meilleure que dans le cas masse-tenseur. Le masse-tenseur est en effet trop mou et il va mettre beaucoup de temps à converger.

5.2 Les contraintes Temps-réel

Afin de tester le temps de calcul qui est la seule vraie valeur importante pour nos conditions temps réel, nous avons évalué plusieurs types de maillages dont vous trouverez la description dans le tableau 5.1. Il est très difficile pour le moment d'évaluer ce qui prend le plus de temps lors de ce calcul car, que se soit pour la première méthode de calcul ou la seconde vus au paragraphe 4.4, beaucoup de paramètres se compensent. Ainsi, dans ce cas la différence de vitesse de résolution entre les maillages avec 2 sous-domaines et les maillages avec 3 sous-domaines est faible. En effet, les sous-domaines étant plus petits dans le second cas, le calcul pour le domaine lui même est plus rapide.

Voici, dans le tableau 5.2, une ébauche de résultat qui correspond au cas le plus rapide, c'est a dire où tous les domaines sont des pré-calculés. On peut y observer pour chaque modèle, sa fréquence de rafraîchissement et sa complexité, exprimées en opérations de base (multiplication d'un tenseur 3x3 par un vecteur 3x1).

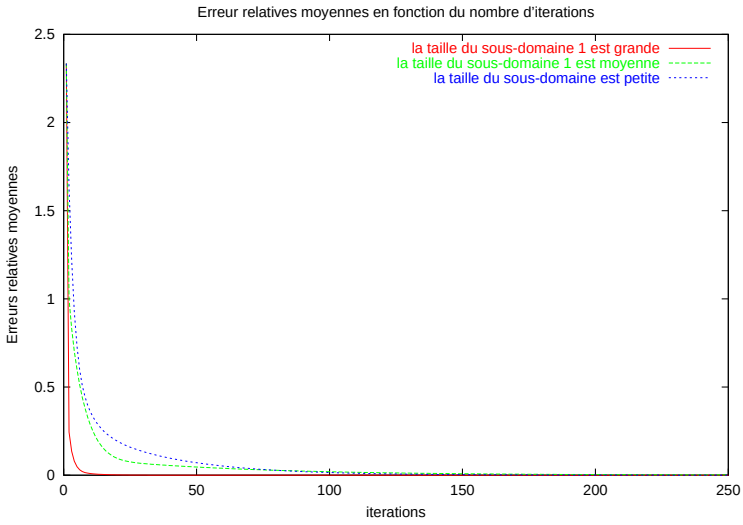


FIG. 5.8 – *Erreurs relatives (erreurs normalisées) lors de l'application d'un déplacement à un maillage ayant 2 sous-domaines mais ayant des tailles de sous-domaines différentes, en fonction du nombre d'itérations.*

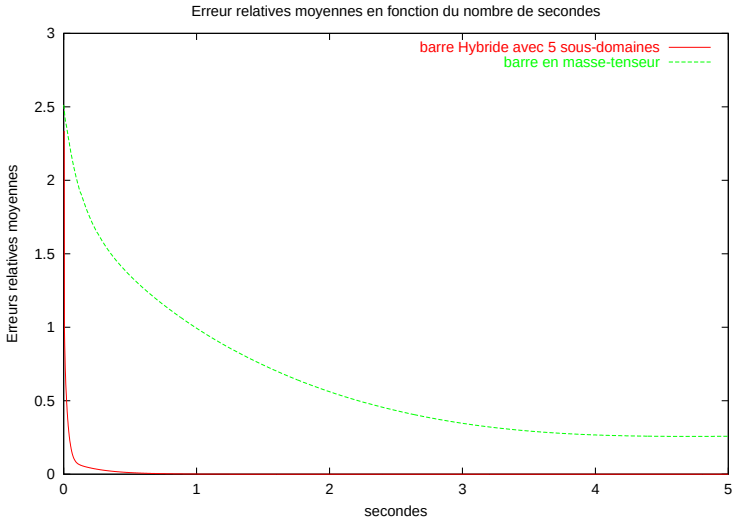


FIG. 5.9 – *Erreurs relatives (erreurs normalisées) lors de l'application d'un déplacement à un maillage hybride ayant 5 sous-domaines et à un maillage masse-tenseur, en fonction du nombre de secondes.*

Maillage	Nb de sommet	Nb moyen de voisin	Descriptif des zones			
barre 2 zones	192	8.6	Zones :	Nb tot	Nb Frontière	Nb Surface
			0	99	10	85
			1	102	10	81
barre 3 zones	192	8.6	Zones :	Nb tot	Nb Frontière	Nb Surface
			0	99	10	85
			1	50	20	30
			2	62	10	41
barre 4 zones	192	8.6	Zones :	Nb tot	Nb Frontière	Nb Surface
			0	99	10	85
			1	40	20	20
			2	40	20	20
			3	42	10	21
barre 5 zones	192	8.6	Zones :	Nb tot	Nb Frontière	Nb Surface
			0	99	10	85
			1	30	20	10
			2	30	20	10
			3	42	10	21
foie500 2 zones	469	11.5	Zones :	Nb tot	Nb Frontière	Nb Surface
			0	278	63	113
			1	253	63	105
foie500 3 zones	469	11.5	Zones :	Nb tot	Nb Frontière	Nb Surface
			0	183	122	38
			1	308	69	119
			2	119	71	30
foie500 4 zones	469	11.5	Zones :	Nb tot	Nb Frontière	Nb Surface
			0	183	128	38
			1	173	127	20
			2	119	71	30
			3	198	61	74
foie4900 2 zones	4919	12.66	Zones :	Nb tot	Nb Frontière	Nb Surface
			0	2863	332	855
			1	2387	332	749

TAB. 5.1 – *Les différents types de maillages utilisés pour les tests et leurs caractéristiques*

Maillage	Masse Tenseur	Pré-calcul de la surface	Pré-calcul tous les noeuds
barre 2 zones	f= 37 Hz C = 1651	f= 37 Hz C = 2.060	f= 37,2 Hz C = 2.066
barre 3 zones		f= 36,4 Hz C = 3.060	f= 36,4 Hz C = 2.720
barre 4 zones		f= 36 Hz C = 3.810	f= 36,2 Hz C = 3.174
barre 5 zones		f= 36 Hz C = 4.460	f= 36,1 Hz C = 3.428
foie500 2 zones	f= 33,6 Hz C = 5393	f= 27,2 Hz C = 29.610	f= 26 Hz C =33.820
foie500 3 zones		f= 25,8 Hz C = 64.349	f= 27 Hz C =52.768
foie500 4 zones		f= 19 Hz C = 93.370	f= 27,5 Hz C =65.886
foie4900 2 zones	f= 7,6 Hz C = 62274	f= 1,2 Hz C = 974.752	f= 0,6 Hz C =1.744.851

TAB. 5.2 – *Vitesses et complexités (en nombre de multiplication de tenseurs 3x3 par un vecteur 3x1) pour chaque maillage de test et chaque méthode.*

5.3 Conclusion

A partir de ces résultats on peut déduire, en ce qui concerne les contraintes temps réels, que le modèle hybride ne remplit pas nos attentes et qu'il se comporte, en terme de fréquences de rafraîchissement, de manière équivalente aux masse-tenseur pour les maillages utilisés dans le simulateur. Cependant il respecte quand même les contraintes temps réels dû au retour visuel.

Par contre, le modèle hybride converge plus rapidement que le masse-tenseur (cf. figure 5.9). Il offre ainsi, pour ce point, une solution intermédiaire entre le masse-tenseur et le pré-calculé. le premier étant très lent, donne un aspect mou à notre foie; alors que le second étant très rapide ne donne pas un aspect assez dynamique (il atteint tout de suite sa position d'équilibre).

Conclusion

Ce stage m'a permis de renouer avec le monde biomédical. J'ai également pu découvrir et apprécier le monde de la recherche avec son état d'esprit, sa passion, son rythme, coupé de réussites et de déceptions. Grâce au travail réalisé par l'équipe EPIDAURE, le réalisme du simulateur va augmenter tout en respectant les contraintes temps réels dû au retour visuel et au retour haptique. Tout ceci est très intéressant et très motivant, surtout lorsqu'on se prend à imaginer l'étendue des applications et des apports au milieu médical.

Même si le modèle hybride ne satisfait pas encore toutes nos attentes, il reste prometteur par sa capacité à être parallélisable et par son temps de convergence, mais il reste un vaste travail encore à réaliser. Il s'agit notamment :

- de finir les tests pour les frontières pré-calculé/masse-tenseur,
- d'implémenter un système permettant de contrôler le nombre d'informations transmises d'une zone à l'autre. Ce contrôle va permettre de compenser le problème de fréquence de rafraîchissement en permettant un compromis entre celle-ci, et le temps de convergence. Ce système ne doit pas affecter les positions d'équilibre du maillage.

Normalement en respectant les objectifs de mon stage, cette étape devrait le conclure. Cependant la cohabitation entre différents modèles physiques va permettre une grande évolution du simulateur en complexifiant ses modèles mécaniques et en augmentant le nombre de nœuds du maillage. On peut donc déjà prévoir quels seront les axes de recherche pour le prochain doctorant qui travaillera sur le simulateur (ce sera peut être moi) :

- Parallélisation du calcul sur chaque sous-domaine afin d'améliorer encore la vitesse du simulateur.
- Introduction de modèles pré-calculés non-linéaires en se basant sur des méthodes algébriques.
- Détection et traitements des contacts surfaces/surfaces avec les structures voisines.

Bibliographie

[1] COTIN, S., (1997). "Modèle Anatomiques Déformables en Temps Réel - Application à la simulation de chirurgie avec retour d'effort". Thèse soutenue le 19 septembre 1997 et effectuée à l'INRIA au sein du projet EPIDAURE.

[2] PICINBONO G., DELINGETTE, H., AYACHE, N.(2001) "Modèles géométrique et physiques pour la simulation d'intervention chirurgicales". Thèse soutenue le 12 février 2001 et effectuée à l'INRIA au sein du projet EPIDAURE.

[3] COTIN, S., DELINGETTE, H., AYACHE, N. (1998). "Efficient Linear Elastic Models of Soft Tissues for Real-Time Surgery Simulation". Rapport de Recherche de l'INRIA numéro 3510.

[4] BRO-NIELSEN, M., COTTIN, S., "Real-Time Volumetric Deformable Models For Surgery Simulation Using Finite Elements and Condensation"

[5] DESIDERI, J.A. (1998), "Modèles Discrets et Schémas Itératifs - *Application aux Algorithmes Multi-grilles et Multidomaines"*, chapitre 7. Édition HERMES

[6] SAAD, Y., (1996), "Iterative Methods for Sparse Linear Systems", chapitre 13. Édition I.T.P.(International Thomson Publishing)

[7] BOUNAIM, A., (1999), "Méthodes de Décomposition de Domaine : Application à la Résolution de problèmes de contrôle optimal". Thèse de doctorat de l'université Joseph Fourier

[8] Site internet de Luc Soler : http://www-sop.inria.fr/epidaure/personnel/soler/

[9] RIXEN, D.,FARHAT, C., "Preconditioning the FETI Method for Problems with Intra- and Inter-Subdomain Coefficient Jumps".

[10] KLAWONN, A.,WIDLUND, OB., "A Domain Decomposition Method with Lagrange Multipliers for Linear Elasticity"

[11] PRESS, W.H., TEUKOLSKY S.A., VETTERLING W.T., FLANNERY B.P., "NUMERICAL RECIPES in C - The Art of Scientific Computing" Second Edition , Cambridge University Press.

[12] DELINGETTE, H.,(2001-2002), "Simulation de Chirurgie", cour du DEA SIC Image-Vision á l'université de Nice Sophia-Antipolis.

[13] MAGNUS, A.,(2001-2002) "Cour d'Analyse Numerique 1A" de l'université Catholiquede Louvain.

[14] MAGNUS, A., (2001-2002) "Cour d'Analyse Numerique 2A" de l'université Catholiquede Louvain.